LY12-5028-2
File No. S370/4300-36

VSE/POWER

Program Product

Program Logic Manual Part 2

Program Number 5666-273

Feature Numbers 6016
6017

Version 2

Third Edition (December 1981)

This is a major revision of and obsoletes LY12-5028-1. Changes or additions to the text and illustrations are indicated by a vertical line to the left of the change. This edition applies to Version 2, Release 1 of VSE/POWER, Program Number 5666-273, together with Version 2, Release 1 of VSE/POWER Shared Spooling feature, Feature Numbers 6016 and 6017 and to all subsequent 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 and 4300 Processors Bibliography, GC20-0001, for the editions that are applicable and current.

It is possible that this material may contain reference to, or information about, IBM products (machines and programs), programming, or services that are not announced in your country. Such references or information must not be construed to mean that IBM intends to announce such IBM products, programming, or services in your country. Publications are not stocked at the addresses given below; requests for copies of IBM publications should be made to your IBM representative or to the IBM branch office serving your locality.

A form for readers' comments is provided at the back of this publication. If the form has been removed, comments may be addressed either to:

International Business Machines Corporation
Department 812BP
1133 Westchester Avenue
White Plains, New York 10604

or to:

IBM Deutschland GmbH
Dept 3248, Programming Publications
Schoenaicher Strasse 220
D-7030 Boeblingen, West Germany

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatever. You may, of course, continue to use the information you supply.

This manual is the second of three volumes describing the internal logic of VSE/POWER. The three volumes are:

- VSE/POWER Program Logic Manual Part 1, LY12-5027
- VSE/POWER Program Logic Manual Part 2, LY12-5028
- VSE/POWER Program Logic Manual Part 3, LY12-5034

This volume (Part 2) describes the method of operation of VSE/POWER itself; Part 3 describes the method of operation of the optional Networking and Remote Job Entry functions.

To use this manual effectively, you should be familiar with the concepts and facilities of VSE/POWER described in the following IBM VSE/Advanced Functions publications:

- VSE/Advanced Functions System Generation, SC33-6096
- VSE/Advanced Functions System Management Guide, SC33-6094
- VSE/Advanced Functions Operating Procedures, SC33-6097
- VSE/Advanced Functions System Control Statements, SC33-6095

RJE,SNA users should also be familiar with ACF/VTAM concepts and facilities as described in:

- ACF/VTAM Concepts and Planning, GC38-0282
- ACF/VTAM Macro Language Reference, SC38-0261

Further VSE/POWER publications are:

- VSE/POWER Installation and Operations Guide, SH12-5329
- VSE/POWER Remote Job Entry User's Guide, SH12-5328
- VSE/POWER Networking User's Guide, SC33-6140
- VSE/POWER Messages, SH12-5520
- VSE/POWER Reference Summary, SH12-5435
- VSE/POWER Shared Spooling User's Guide, SH12-5330

CONTENTS

LIST OF ILLUSTRATIONS

METHOD OF OPERATION

This manual describes in detail the phases of the VSE/POWER program with the exception of those concerning remote job entry. An introduction describes linkage conventions between routines. Remote job entry phases and PNET phases are described in VSE/POWER Program Logic Manual Part 3, LY12-5034.

Modules IPW$$MD and IPW$$MM are not documented, due to their simplicity. They contain only VSE/POWER message definitions, and otherwise no executable code.

VSE/POWER LINKAGE CONVENTIONS

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

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

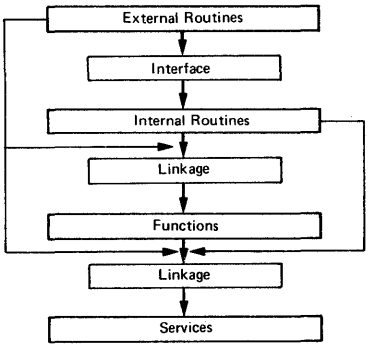


Figure 1. Hierarchic Structure of VSE/POWER

Register Conventions

This section describes the standard functions and uses assigned to certain of the general purpose registers throughout VSE/POWER. The VSE/POWER 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 VSE/POWER partition at all times during VSE/POWER 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 VSE/Advanced Functions 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 - General Use

Registers 4-9 are available for general task use.

Interface Linkage

Each external and internal routine of VSE/POWER 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).

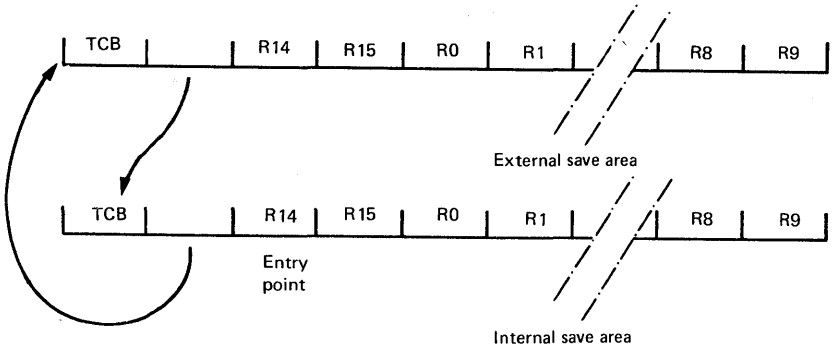


Figure 2. Relationship of Internal and External Save Area

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.

Function Linkage

Each VSE/POWER 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 VSE/POWER 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 3shows 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$WFQ			list						R12=return
IPW$WFC			CCB or						address
IPW$WFS			ECB						
IPW$WFD									
IPW$RSR					return		RCB		
IPW$RLR					return		RCB		
IPW$RSW	function-code	real address	size	virtual address	return				
IPW$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$VDA		return code			return				R6=PDB R8=CCW addr.
IPW$GAM	destid or zero		message number	Pointer to message	return		msg area or zero		
IPW$SRm	request code		remote id		return				
IPW$RSV	function-code		length	address or area or zero	return				R4=Pointer to anchor
IPW$RLV	function code		address of area or zero		return				R4=Pointer to anchor
IPW$UNV	address of anchor to queue		address of area or zero	address of area or zero	return				R4=Pointer to anchor
IPW$GTE	length of trace area			address trace area	return				
IPW$STm			address TQE		return				
IPW$NTY	pointer to target node name		address NMR		return				

Figure 3. Contents of Registers When a Service is Invoked

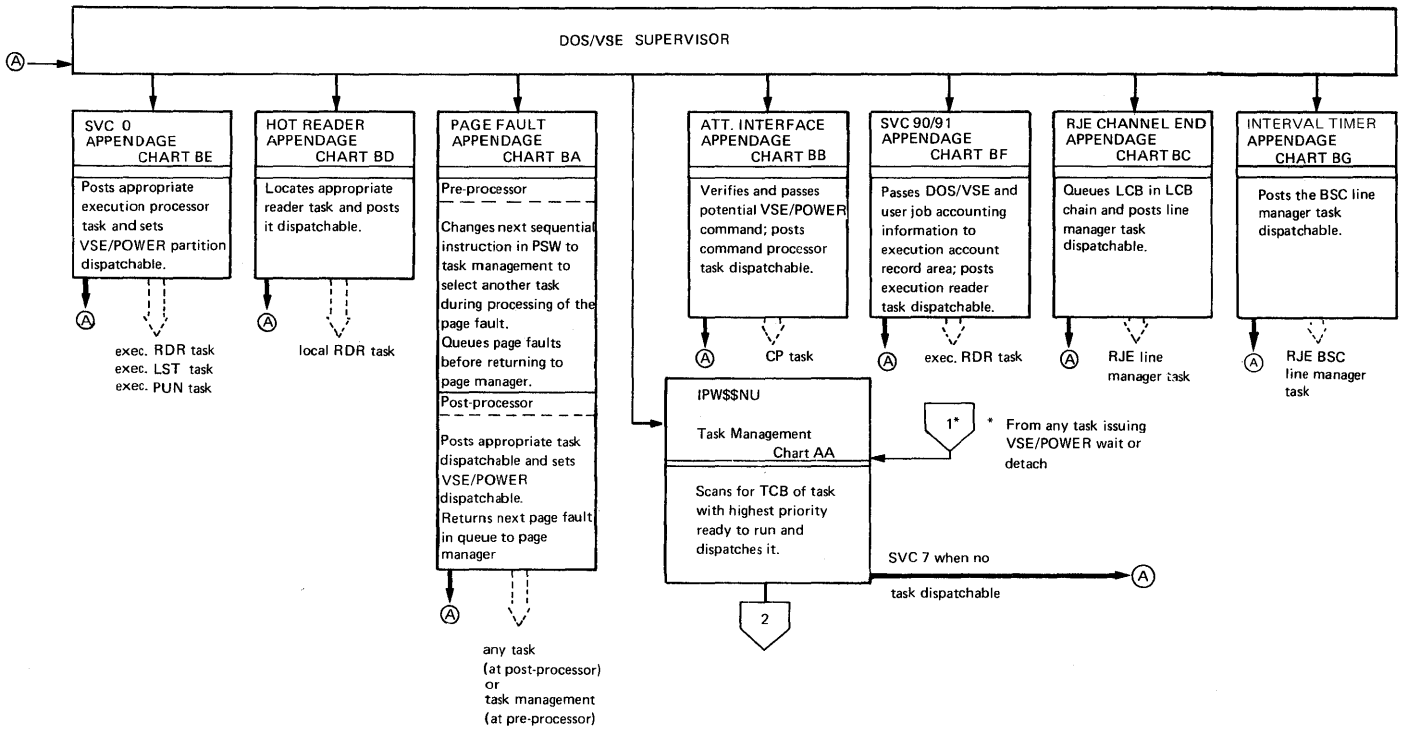


Figure 4. Interfaces and Task Structured References (Part 1 of 5)

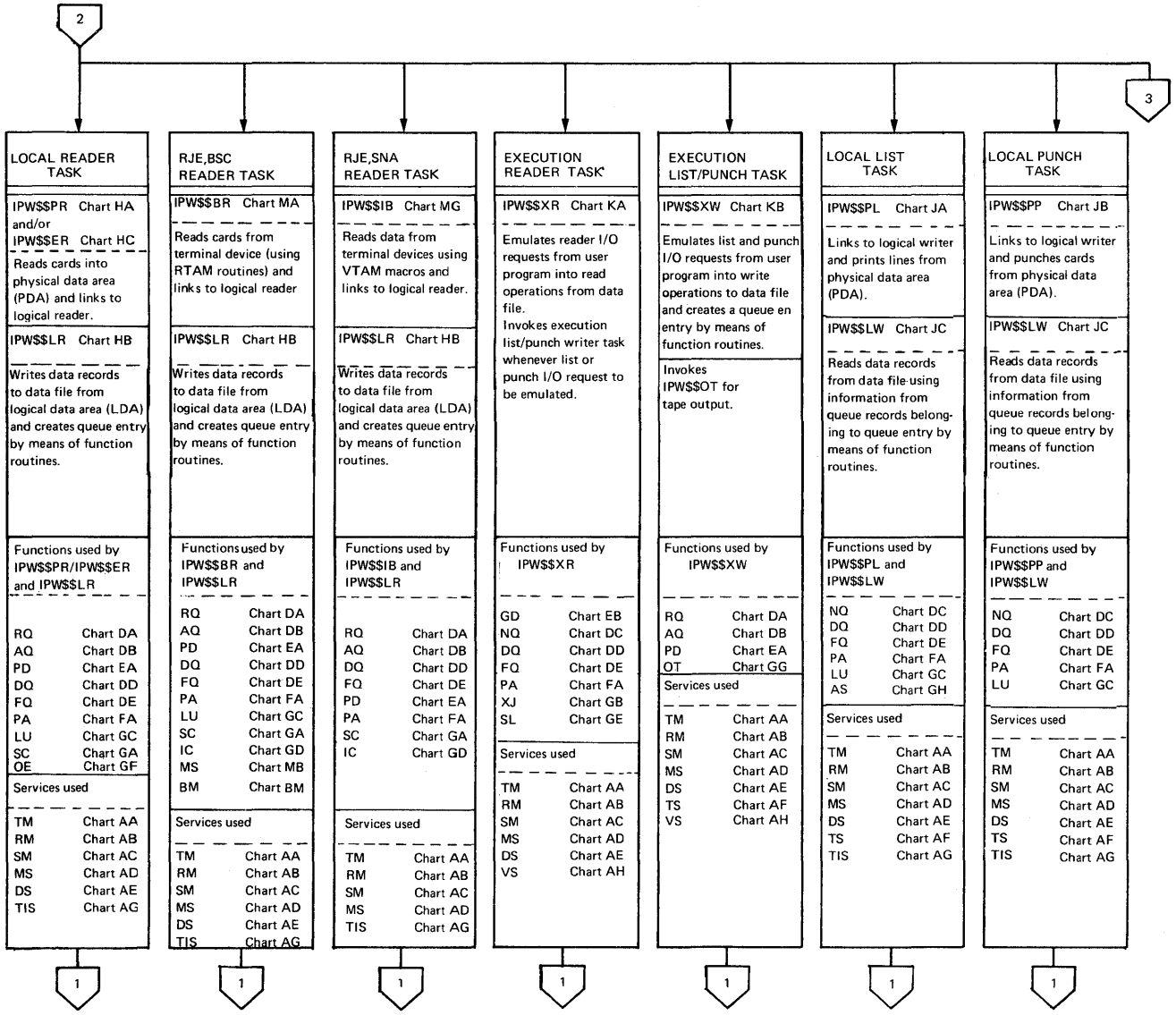


Figure 5. Interfaces and Task Structured References (Part 2 of 5)

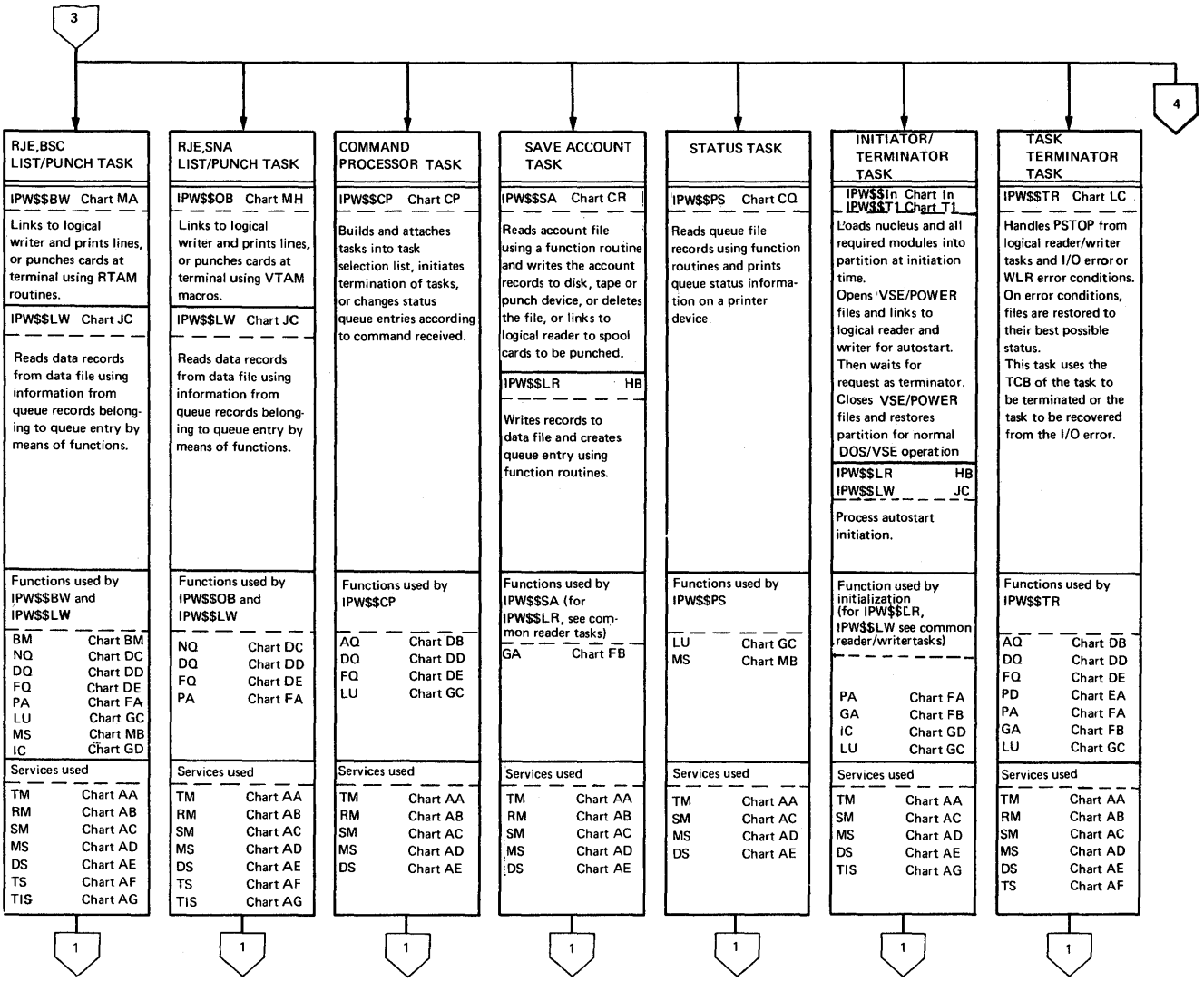


Figure 6. Interfaces and Task Structured References (Part 3 of 5)

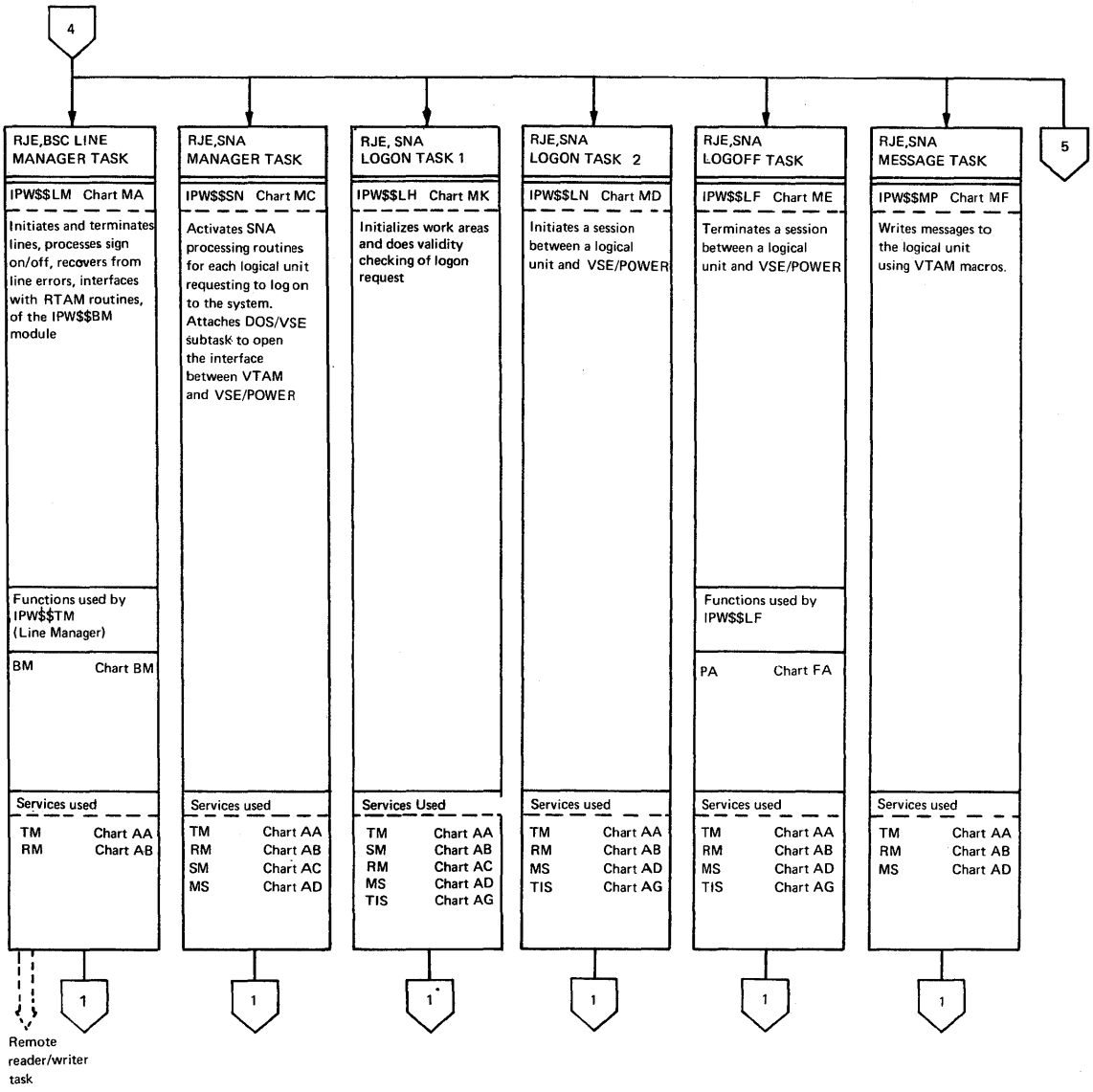


Figure 7. Interfaces and Task Structured References (Part 4 of 5)

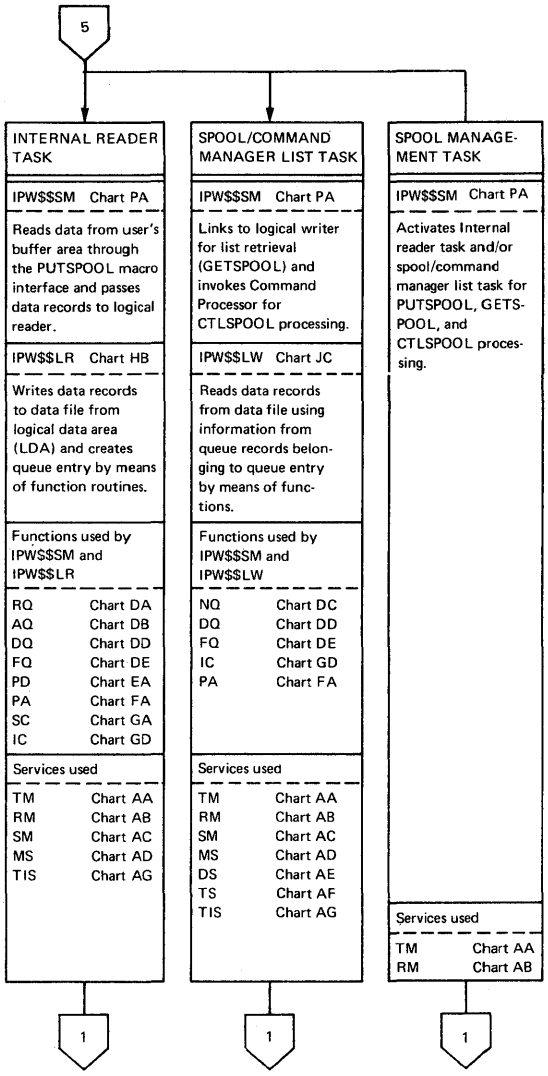


Figure 8. Interfaces and Task Structured References (Part 5 of 5)

UNDERSTANDING THE PHASE DESCRIPTIONS

Parts 2 and 3 of the VSE/POWER program logic manual contain detailed descriptions of the VSE/POWER phases. The phases are described using two methods of presentation.

- Text description

  Some phases are presented in the form of charts containing a written description of the phase. An explanation of what these charts contain and an example are shown in Figure 9.

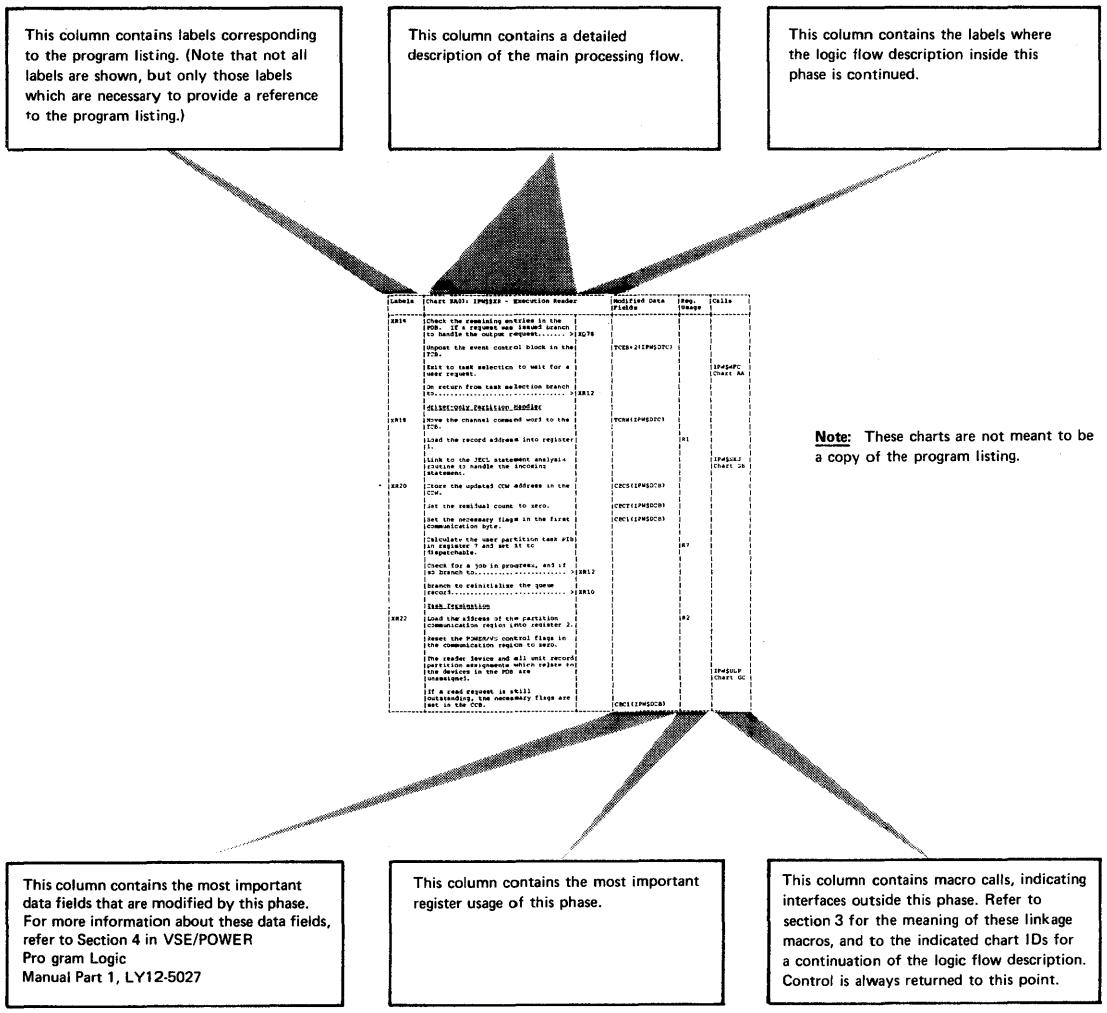


Figure 9. Explanation of Text Phase Descriptions

- Machine-drawHIPO Charts

  This is a diagrammatic form of phase description consisting of machine-drawn HIPO diagrams. HIPO stands for Hierarchy plus Input-Process-Output, which is a program documentation technique.

  For each phase, the first chart is an overview of the phase segments.

  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.

  Note:

  - Column 'Module' contains references to VSE/Advanced Functions macros that are used at this point in the code to provide linkage to VSE/Advanced Functions functions or services.

- Column 'Label' contains cross-references to source-code listings (microfiche) to help you find this location in the code quickly.
- Column 'Reference' contains the abbreviated name of a VSE/POWER internal macro - for example, '$RLW' refers to macro 'IPW$RLW'. For more information about such linkage macros, refer to Section 4 in VSE/POWER Program Logic Manual Part 1, LY12-5027.

The charts are generated by the program HIPODRAW and the symbols used are explained in Figure 10.

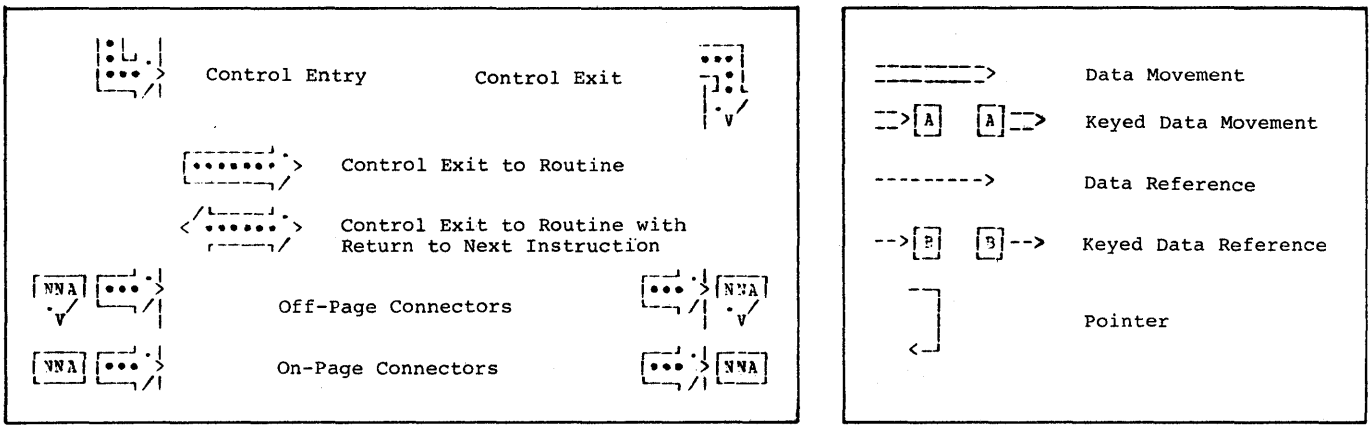


Figure 10. Symbols Used in HIPODRAW Charts

METHOD OF OPERATION: CHARTS

IPW$$AQ - Add Queue Set To Chain	
Label	**Routine**
AQ00	Function Entry
AQ10	Examine Class Table
AQ50	Add to Start of Chain
AQ55	Add to Middle of Chain
AQ60	Add to End of Chain
AQ65	Add Complete Chain
AQ70	Post Class Table ECB
	Post Network Transmitter (Local/Shared Support)
AQ7S	Post Shared System SYSID
AQ73	Post RJE Writer
AQS0	Post Shared System RJE Writer
AQ88	Function Exit

Services Used	
Service	**Macro**
Resource Management	IPW$RLR IPW$RSR
Disk / Tape Service	IPW$CTT IPW$RDQ IPW$WTQ IPW$WTT
Notify Service	IPW$NTY

Called by	
Module	**Description**
IPW$$CA	Command Processor (PALTER)
IPW$$CR	Command Processor (PRELEASE)
IPW$$LR	Logical Reader
IPW$$NR	PNET Receiver
IPW$$OF	Offload Queues
IPW$$TR	Task Terminator
IPW$$XW	Execution Writer

Labels	Chart AQ: IPW$$AQ - Add Queue Set to Chain		Modified Data Fields	Reg. Usage	Calls
AQSD	The first 16 bytes constitute the section descriptor:				
	'AQCS release'				
	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$WTT
	• Write two tapemarks, backspace two				IPW$CTT
	• Exit.....>	AQ89			
AQ06	Set addressability disk management block (DMB).			R6	
	Lock DMB.				IPW$RSR
	If the queue record traps are active and the record is not flagged as free, force program check.....>	AQSD			
AQ07	If 'KEEP' is not specified branch to.....>	AQ08			
	Write queue record back and branch to.....>	AQ10			IPW$WTQ
AQ08	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$WTQ
	Read first record back into the auxiliary queue area.		QCQW(IPW$DQC)		IPW$RDQ
	Copy information in queue record area				
	• Overwrite record pointers		QRCF(IPW$DQR)		
	• Overwrite control seek address		TCQW(IPW$DTC)		
	• Reset first-in-set switch		QRFS(IPW$DQR)		

Labels	Chart AQ: IPW$$AQ - Add Queue Set to Chain		Modified Data Fields	Reg. Usage	Calls
	Examine Class Table				
	If the queue record PNET target node				
	is the local system.....>	AQ18	QRTN(IPW$DQR)		
	Address the correct XMT queue entry				
	(either JOB or OUTPUT)				
	If add queue set with keep.....>	AQ70			
	If PNET feature not available.....>	AQ22			
	If PNET target node is known.....>	AQ22			
	Else the node is indicated as unknown		TCF3(IPW$DTC)		
	DISP=H is set. Branch.....>	AQ22	QRDP(IPW$DQR)		
AQ18	If the target userid is specified in				
	the format 'Rnnn', then it is convert				
	-ed to hex and stored in the remote				
	'to' field.		QRTJ(IPW$DQR)		
AQ19	Address class table entry:				
AQ22	If pointers are zero add complete				
	chain.....>	AQ65			
	If add queue set with keep.....>	AQ70			
	Translate relative disk address of		QCQW(IPW$DQC)	R14	
	last in class pointer.....>	AQ90			
	• Read last-in-class record in				IPW$RDQ
	auxiliary queue area				
	• Check priority for add to end of				
	chain.....>	AQ60			
AQ25	Scan chain backwards, until priority		QCQW(IPW$DQC)		IPW$RDQ
	of new set is equal or lower for add				
	to middle of chain (auxiliary area				
	now contains previous queue record) >	AQ55			

Labels	Chart AQ: IPW$$AQ - Add Queue Set to Chain		Modified Data Fields	Reg. Usage	Calls
AQ50	Add to Start of Chain				
	If this is same queue set, exit.....>	AQ70			
	Translate and store first-in-class pointer.....>	AQ95	CTQF(MCT)	R14	
	Set previous class pointer to zero.		QRQP(IPW$DQR)		
	Set forward pointer to next-in-class.		QRQN(IPW$DQR)		
	Write new record.				IPW$WTQ
	Reset backward pointer to first-in-class.		QCQP(IPW$DQC)		
	Write back current first-in-class record from auxiliary area.				IPW$WTQ
	Branch to completion.....>	AQ70			
AQ55	Add to Middle of Chain				
	If this is the same queue class, exit.....>	AQ70			
	Set backward pointer to previous-in-class.		QRQP(IPW$DQR)		
	Set forward pointer to next-in-class.		QRQN(IPW$DQR)		
	Write new record.				IPW$WTQ
	Reset forward pointer to next-in-class.		QCQN(IPW$DQC)		
	Write previous record back.				IPW$WTQ
	Read next record.				IPW$RDQ
	Reset backward pointer to next-in-class.		QCQP(IPW$DQC)		
	Write next record back.				IPW$WTQ
	Branch to completion.....>	AQ70			
AQ60	Add to End of Chain				
	If this is same queue set, exit.....>	AQ70			
	Translate and store last-in-class pointer.....>	AQ95	CTQL(MCT)	R14	
	Set forward class pointer to zero.		QRQN(IPW$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$DQR)		
	Write back old record.				IPW$WTQ
	Branch to completion.....>	AQ70			

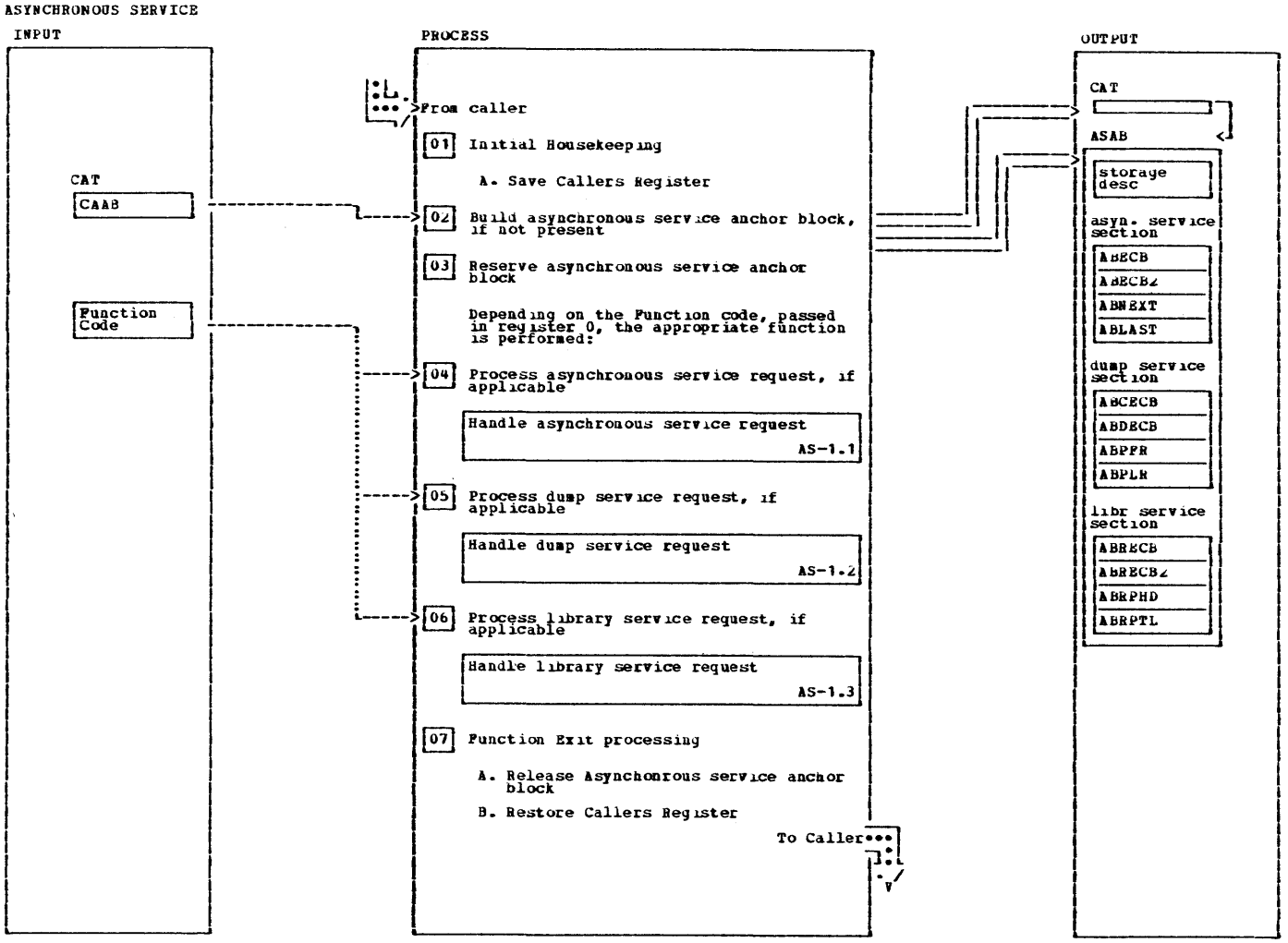
Labels	Chart AQ: IPW$$AQ - Add Queue Set to Chain		Modified Data Fields	Reg. Usage	Calls
AQ65	Add Complete Chain				
	Translate and store first-in-class..>	AQ95	CTQF(MCT)	R14	
	Last-in-class pointers in MCT.		CTQL(MCT)		
	Set backward class pointer to zero		QRQP(IPW$DQR)		
	Set forward class pointer to zero.		QRQN(IPW$DQR)		
	Write new record.				IPW$WTQ
	Branch to completion.....>	AQ70			
	Function Exit				
AQ70	If the queue set is added for a reader task, branch to.....>	AQ71			
	If the queue set is added for an RJE task, branch to.....>	AQ73			
AQ7A	If DISP='L' or DISP='H', branch.....>	AQ88			
	POST class table entry ECB If the PNET feature is not available or the queue record is not for the XMT queue, branch.....>	AQ7S	CTQL(IPW$DCT) CAPN(IPW$DCT)		
	If the target node is unknown.....>	AQPX	TCF3(IPW$DTR)		
	If the target node primary routing is valid.....>	AQP4	NDTPR(PNODE)		
	If an alternate route is either not specified or is invalid,.....>	AQPX			
	Otherwise reserve the PNCB and branch	AQP8			IPW$RSR
AQP4	If a transmitter task exists, branch>	AQP6	PNCBNCB(IPW$DPN)		
	Otherwise, BAL to post the shared system primary node (NAT).....>	AQPS		R8	
	If the posting was successful.....>	AQPR		R0	
	Otherwise BAL to post the shared system alternate node.....>	AQPS			
	then branch.....>	AQPR			
AQP6	If a transmitter task is available for the primary node, branch.....>	AQP8			
	Otherwise, branch.....>	AQP9			
AP8	If the transmitter is not signed on.>	AQP9			
	Otherwise BAL.....>	AQPT			
	and then branch.....>	AQPR			
AQP9	BAL to attempt to post the NAT .....>	AQPS			
	If the posting was successful,.....>	AQPR			
	Otherwise, if no alternate route was specified, or it was invalid.....>	AQPR			
AQP8	If no transmitter tasks exist, or none are serving the alternate node, then branch.....>	AQPD			
	Otherwise if the transmitter is signed on, then .....>	AQPE			
AQPD	BAL to attempt to post the NAT .....>	AQPS			
	then branch.....>	AQPR			
AQPE	BAL to post the transmitter .....>	AQPT			
AQPR	Release the PNCB				IPW$RLR
AQPX	Exit	AQ88			

Labels	Chart AQ: IPW$$AQ - Add Queue Set to Chain		Modified Data Fields	Reg. Usage	Calls
	PNET Post a Transmitter				
AQPT	Search the specified node transmitter table for the given queue type for an available transmitter				
	If found indicate task creation, and post the line driver		NCBEST1(IPW$DNC) NCBACT1(IPW$DNC)		
	Return to caller			R8	
	Post a Shared System Node (NAT)				
AQPS	If the system is not shared .....>	AQPSE	MRSO(IPW$DQC)		
	If the specified node is signed on at some other shared system, then post the given queue type		NATFLAG		
	Otherwise branch.....>	AQPSE			
	Set return code to no error			R0	
AQPSX	Return to caller			R8	
AQPSE	Set error return code, branch.....>	AQPSX		R0	
AQ7S	If a job is put in one of the VSE/POWER queues, requesting processing by a specific system is done by means of the SYSID parameter. The corresponding bit representing the class in the SYSID-class-table which exists for each SYSID in the master record, is posted. Branch.....>	AQ88			
	Else branch.....>	AQ88			

Labels	Chart AQ: IPW$$AQ - Add Queue Set to Chain		Modified Data Fields	Reg. Usage	Calls
	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 the address of list or punch task in R4.			R4	
AQ76	Load the address of output DCT.			R3	
	Loop to find the corresponding DCT. If no matching DCT found branch to..>	AQ88			
AQ77	If the device is stopped branch to..>	AQ88			
	Load address of class list.				
	If the class in the DCT matches that in the queue record, set the output indicator.		LCBOUSW		
	Branch to .....>	AQ88			
AQ80	Check whether RJE,SNA support has been specified at VSE/POWER generation. If not, branch to exit >	AQS0			
	Lock the SNA control block.				IPW$RSR
	Establish addressability for the SNA unit control block chain in register 3.			R3	
	If no SNA unit control block is present, branch to.....>	AQ87			
AQ81	Scan the SNA unit control block chain for an SNA unit control block associated with the TO terminal user. If not found, branch to.....>	AQ87			
AQ83	Point to first (next) device entry. If device is RDR, branch to.....>	AQ87			
	If devices do not match with QRQ1, then branch to.....>	AQ83			
	If device is not started or device not available.....>	AQ83			
	Scan for matching class. If not found, branch to.....>	AQ83			
AQ86	Set device not available and output available.		SUL1(IPW$DSU)		
AQ87	Unlock the SNA control block.				IPW$RLR

Labels	Chart AQ: IPW$$AQ - Add Queue Set to Chain		Modified Data Fields	Reg. Usage	Calls
AQ88	Check whether the TCB belongs to a command processor task or to a save-account task. If so, branch to.....>	AQPM0			
	Unlock the disk management block.				IPW$RLR
AQPM0	If target node was unknown .....>	AQ89			
	Otherwise reset 'unknown' indicator			R4	
	Issue local operator message 'RA1I'				IPW$GAM
	If notify was requested then send the message to the 'to-be-notified' user		QRON(IPW$DQR) QROU(IPW$DQR)		IPW$NTY
AQ89	Set the function track byte in the TCB to C'E' to indicate processing complete.		TCFT(IPW$DTC)		
	Return to calling routine.	R14			IPW$RET
	If the queue/data file is shared, then the appropriate bits are set in the remote table to signal other processors that work is available.				
AQS0	If the queue/data file is not shared, branch to.....>	AQ88			
	Calculate the displacement in the remote table by dividing the remote-id by eight. The remainder is the bit that will be posted.		SSRT		

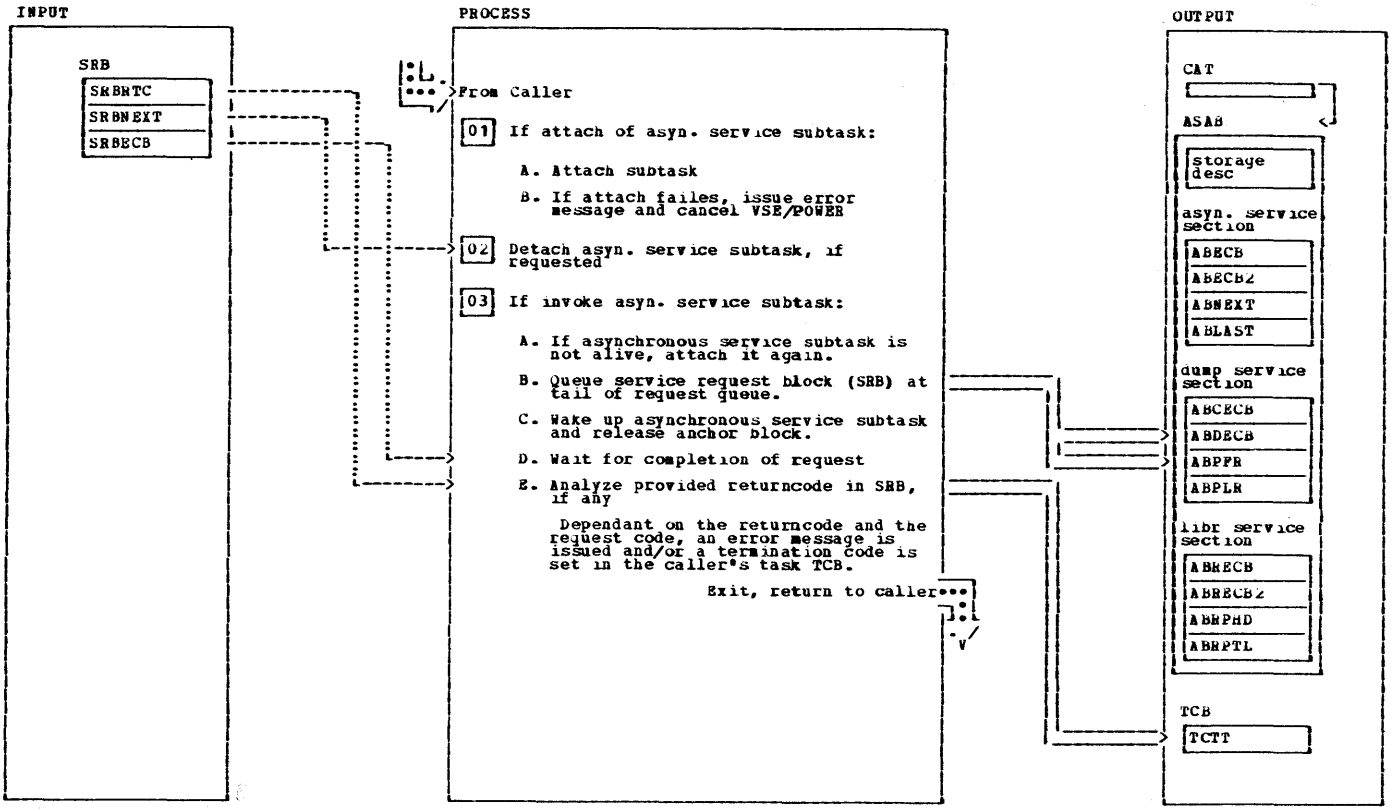
Labels	Chart AQ: IPW$$AQ - Add Queue Set to Chain		Modified Data Fields	Reg. Usage	Calls
	Relative to Absolute Seek Address Conversion Subroutine				
AQ90	The relative queue record address passed in register 1 is converted to an absolute seek address which is stored in the eight byte receiving field addressed by register 2. For FBA devices it is not necessary to convert from relative to absolute, because READ/WRITE will be done with relative block number only. If FBA device is used, store relative block number (R2) in register 1 (relative block number). Return to caller via register 14.	R14		R0,R1, R3	
	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. For FBA devices it is not necessary to convert from absolute to relative, because READ/WRITE will be done with relative block number only. If FBA device is used, store absolute block number (R2) in register 1 (relative block number). Return to caller via register 14.	R14		R0,R1	



NOTES	MODULE	LABEL	REF
1A The caller's registers are saved in the save area addressed by register 13		AS000	$SAV
2 A check is made if there is already an Anchor Block. If not, Space is acquired and the control block is initialized with following values: - storage descriptor			$RSW
2 The address if the asynchronous service anchor block is stored into the CAT			
3 The asynchronous service anchor block is reserved for the duration of the appropriate function. The IPW$IAS macro has set up following function codes: - x'00' invoke asyn. service function - x'04' attach asyn. service subtask - x'08' detach asyn. service subtask - x'0c' invoke dump subtask - x'10' attach dump subtask - x'14' detach dump subtask - x'18' invoke library service subtask - x'1c' attach library service subtask - x'20' detach library service subtask		AS005	$RSR

NOTES	MODULE	LABEL	REF
The function code is used as a branch index to branch to the appropriate routine.			
7A 7B The caller's reagister are re-stored from the the save area addressed by register 13		AS520	$RLR $RET

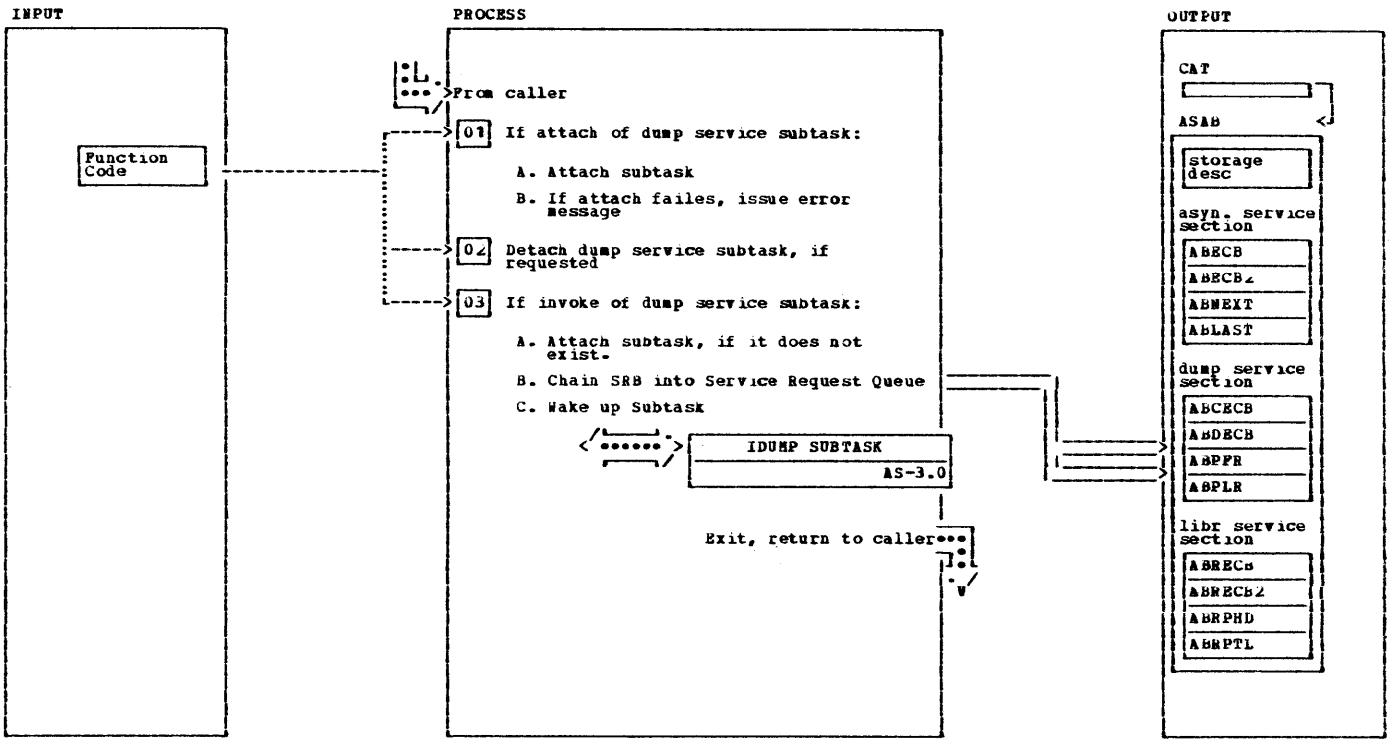
Handle asynchronous service request



	NOTES	MODULE	LABEL	REF
1	The asynchronous service subtask is attached. If the attach failes, because no subtask available anymore, message 1QA0I NO SUBTASK AVAILABLE FOR ttttt,cuu is issued and VSE/POWER is abnormally terminated.	ATTACH	AS100	$GAM
1B	Note: It is most unlikely, that the ATTACH failes, because the request in done at VSE/POWER initialization time.			$CNC
2	If the subtask still exists, the detach is performed and the task is put in wait state until the detach is completed.	DETACH	AS200	
3A	The subtask communication ECB is examined, if the subtask still exists. If not, the attach function is called internally.		AS300	

	NOTES	MODULE	LABEL	REF
3B	T SRB is chained as last entry in the service request queue.			
3C		POST		$RLR
3D	The asynchronous service subtask posts the ECB supplied in the SRB when the request has been either sucessfully or unsucessfully completed			
3E	One of following messages can be issued: 1Q54I FCB ERROR FOR jjjjjjjj nnnnn ttttt,cuu 1QA1I SETPRT ROUTINE NOT FOUND IN SVA ttttt,cuu 1QA3I SETPRT ERROR FOR jjjjjjjj nnnnn ttttt,cuu 1QA4I OUTPUT PROCESSING STOPPED jjjjjjjj nnnnn ttttt,cuu 1QA6I NO STORAGE AVAILABLE FOR ttttt,cuu			$GAM

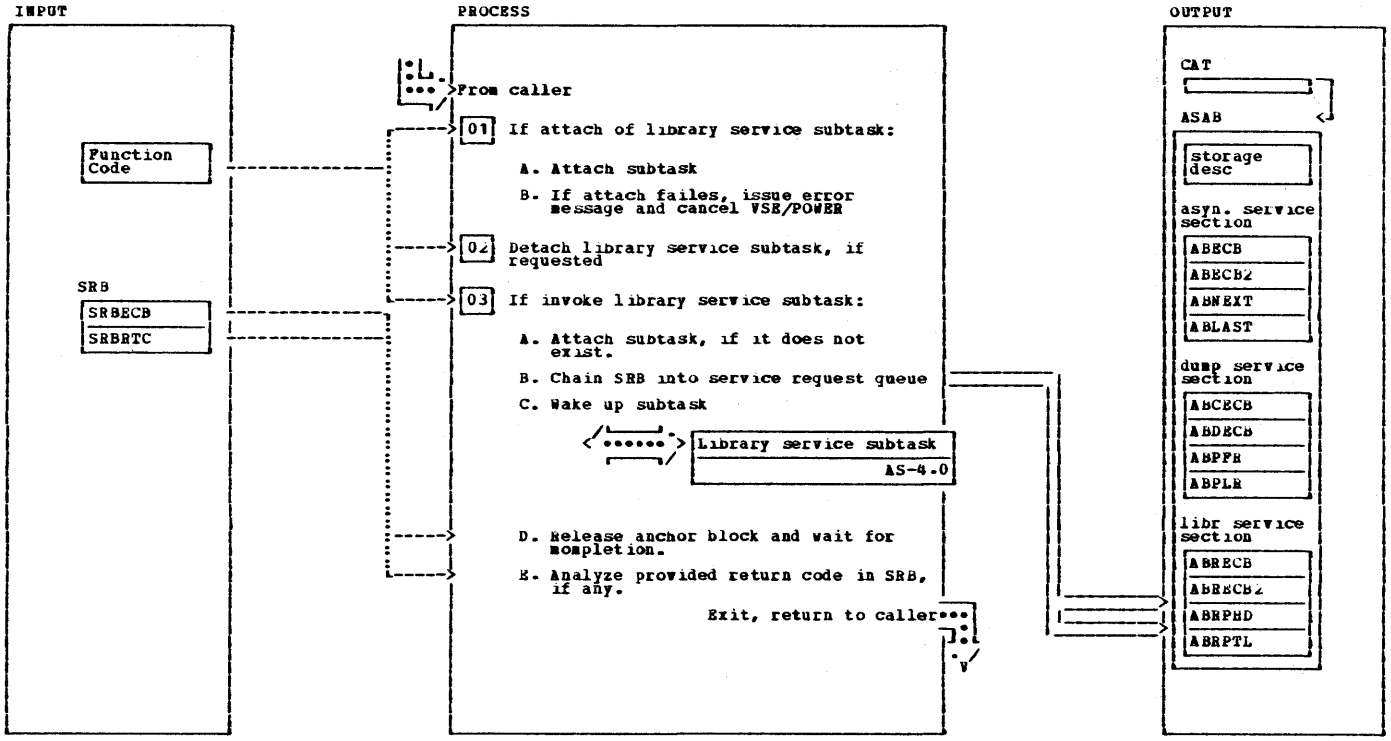
Handle dump service request



	NOTES	MODULE	LABEL	REF
1	The dump service subtask is attached. If the attach failes, because no subtask available anymore, message IQA0I NO SUBTASK AVAILABLE FOR ttttt,cuu is issued	ATTACH	AD100	$GAM
2	If the dump subtask does not exist anymore, no detach is done.	DETACH	AD200	

	NOTES	MODULE	LABEL	REF
3A	The Subtask ECB is examined if the Subtask is still alive. If not, the Attach Function is automaticaaly invoked.			
3B	The Request will be chained.			
3C	The Subtask is waked up.	POST		

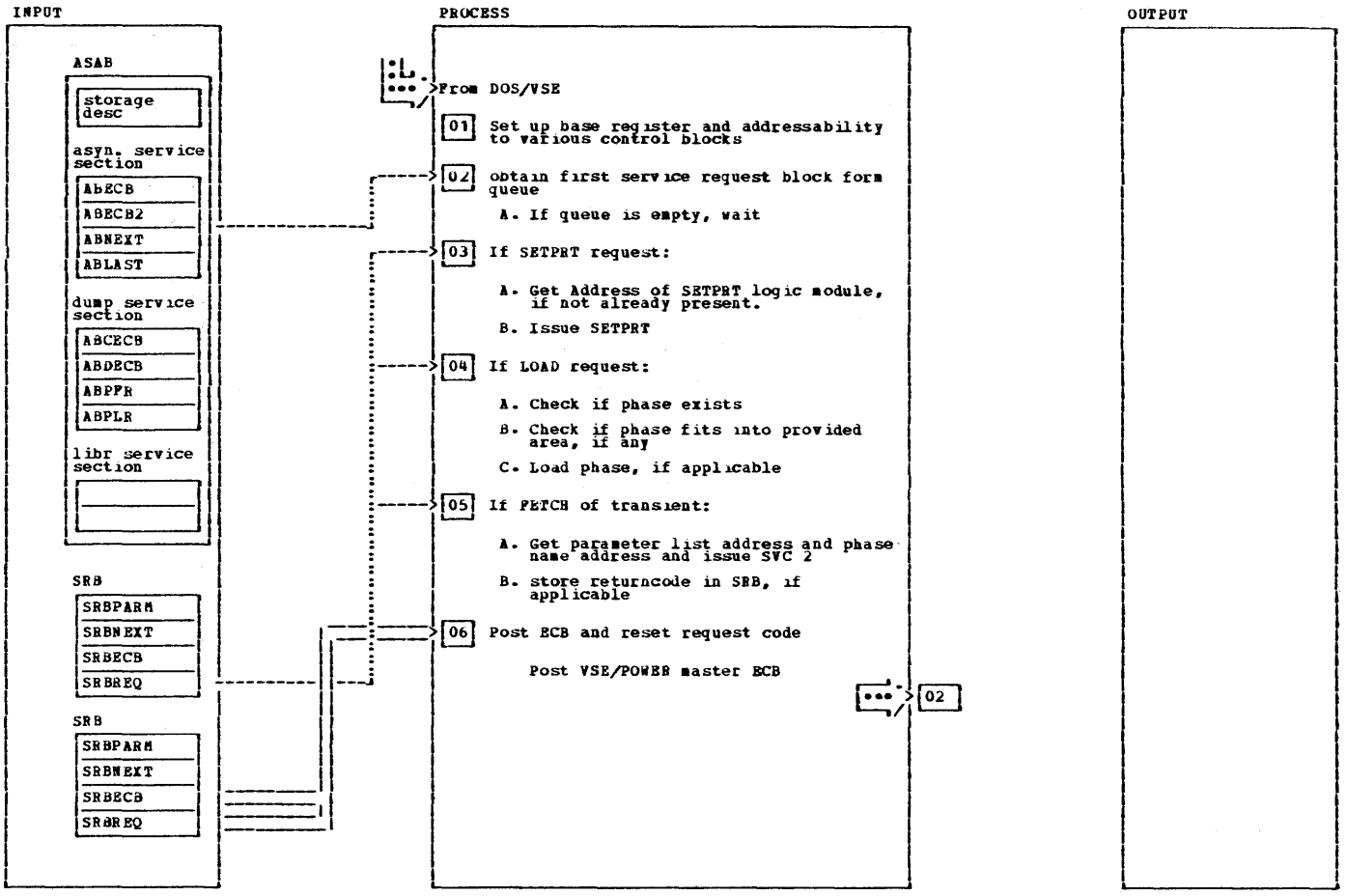
Handle library service request



NOTES	MODULE	LABEL	REF
1 The library service subtask is attached. If the attach failes, because no subtask available anymore, message 1Q10I NO SUBTASK AVAILABLE FOR ttttt,cuu is issued and VSE/POWER is abnormally terminated.	ATTACH	lb100	$GAM
1B Note: It is most unlikely, that the ATTACH failes, because the request in done at VSE/POWER initialization time.			$CNC
2 If the subtask still exists, the detach is performed and the task is put in wait state until the detach is completed.	DETACH	lb200	
3A The subtask ECB is examined if the subtask is still alive. If not, the ATTACH function is invoked.		LB300	

NOTES	MODULE	LABEL	REF
3B The request will be chained.			
3C The subtask is waked up.	POST		
3D The library service subtask posts the ECB supplied in the SRB when the request is either successfully or unsuccessfully completed.			$RLH $WFC
3E One of the following messages can be issued: 1Q44I BOOK s.bbbbbbbb NOT FOUND. 1Qc0I ERROR DURING SOURCE LIBRARY ACCESS, RC=nnnn			$GAM

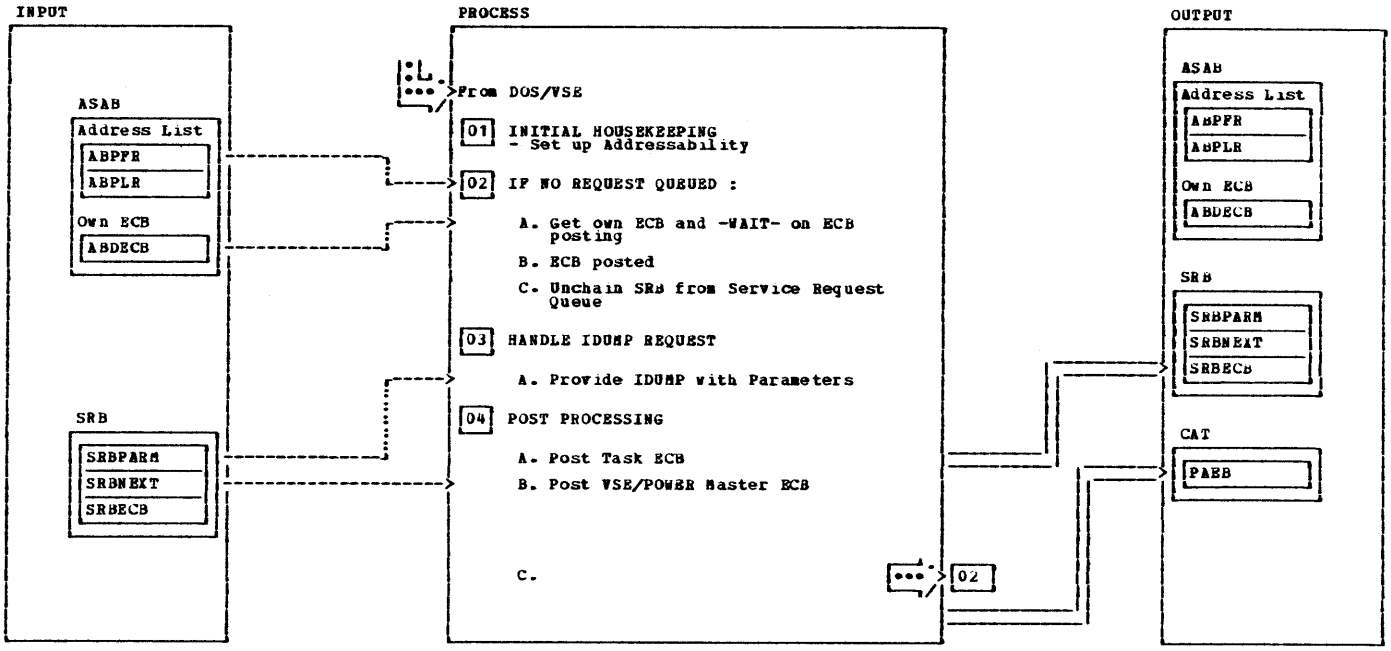
Asynchronous service subtask



	NOTES	MODULE	LABEL	REF
2	If no SRB is in the queue, the subtask is put into wait state until waked up by asynchronous service. Depending on the request code, set up in the service request block, the appropriate function is performed: - SETPRT request - LOAD request - FETCH transient	WAIT	ST010	
3			ST060	
3A	If this is the first SETPRT request, a LOAD with the TXT-NO option is issued, to get the address of the SETPRT logic module (IJVSPRDV) in the SVA. If the phase is not found nor in the SVA, return code 'found not ' is set in the SRB and the request is ignored. Otherwise the address of the SETPRT module is stored in the asynchronous service anchor block.	LOAD		

	NOTES	MODULE	LABEL	REF
3B	The SETPRT parameter list, wich address is stored in the SRB, is picked up and used to perform the SETPRT. On return, the return code is stored into the SRB.	SETPRT		
4	A LOAD with TXT=NO option is issued to check if the phase to be loaded exists. If not, an appropriate returncode is set in the SRB and a branch to the exit is made. Otherwise the phase size is calculated and checked against the maximum size specification obtained from the SRB, if one is present. Additionally, the actual phase size is stored into the SRB. If a load address was specified in the SRB, the phase is loaded at that place.	LOAD	STLOAD	

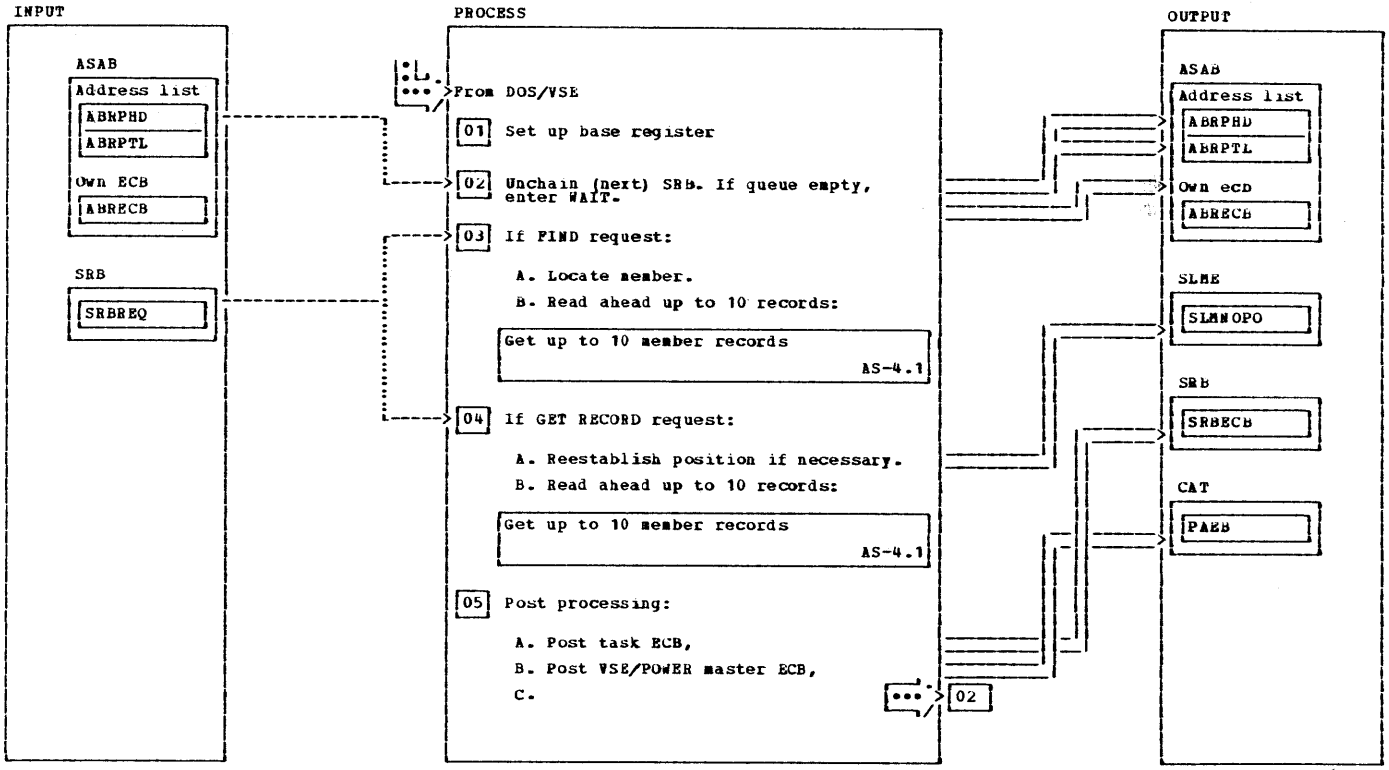
Asynchronous dump subtask



NOTES	MODULE	LABEL	REF
2 The Pointer ABFPR is examined if there is a Request queued (ABFPR points to 1st Request).		IDS05	
2A Task is set in Wait State until a Request is queued that means a wait on ECB posting.	WAIT		
2B After the Task is waked up, return is made to the Subtask Entrypoint.			

NOTES	MODULE	LABEL	REF
3 The SRBPARM Field in the SRB contains all the needed Informations for the IDUMP (e.g.Start/End Addr.)	IDUMP	IDS30	
4 On return from IDUMP the Task ECB and the VSE/POWER Master ECB are posted.			

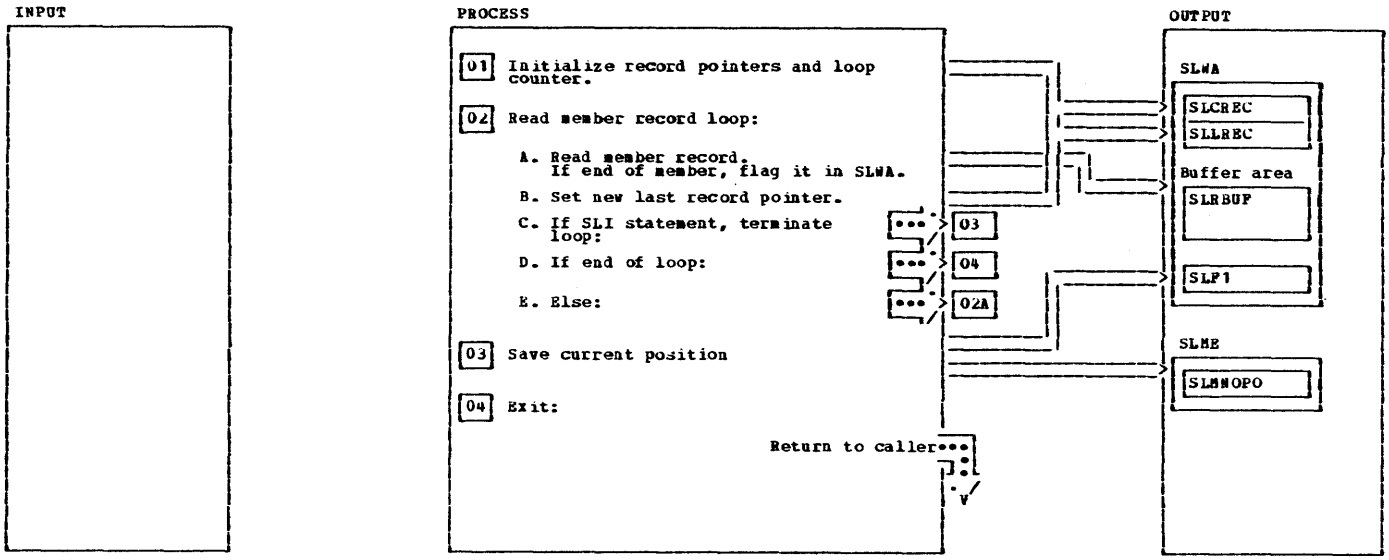
Library service subtask



NOTES	MODULE	LABEL	REF
2 The pointer ABRPHD is examined if there is a request queued (ABRPHD points to the 1st request). If the queue is empty, a wait for further posting by the main task is entered. If the task is waked up, above check is done again. If the queue is not empty, the first request block is dequeued and addressability of the SLWA and the SLME is set up using the pointers passed with the just unchained SRB.	WAIT	LBSM05	
3A The FNDSL macro is issued to locate the member described by the SLME of the current request. If the member could not be found, a flag is set in the SRB and the post processing is entered.	FNDSL	LBSF00	

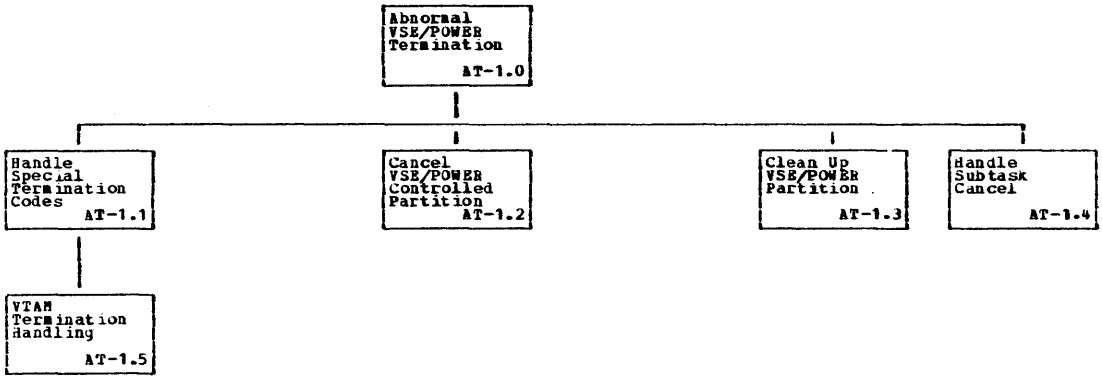
NOTES	MODULE	LABEL	REF
4A A flag in the SLME of the current request indicates, if a member position must be reestablished by a POINT request. POINT is only required in case of nested SLI processing if an inner nesting level is left at end of member, and processing is resumed on the next outer nesting level.	PTSL	LBSG00	

Get up to 10 member records

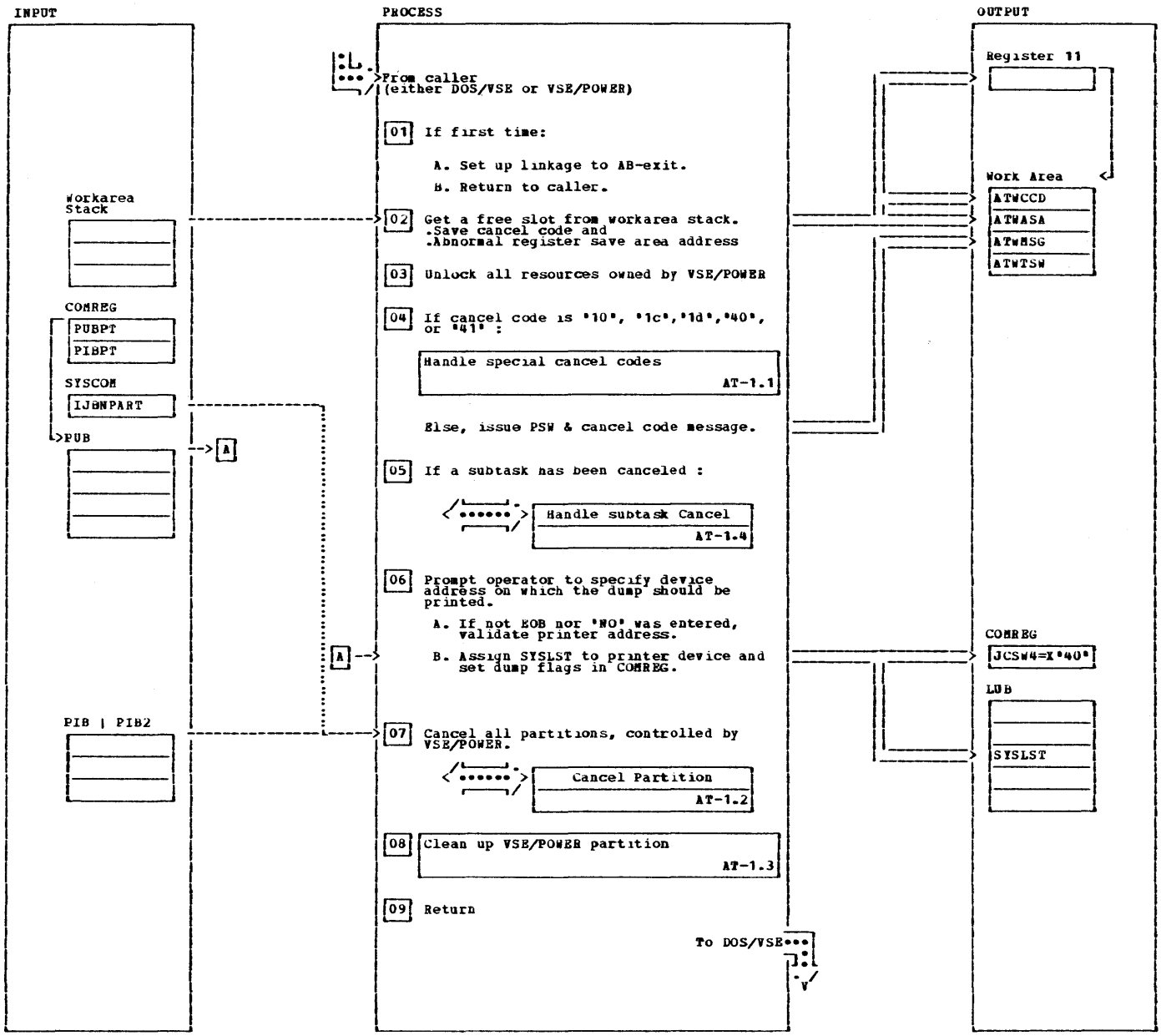


NOTES	MODULE	LABEL	REF
1 The current record pointer is set to the begin of the internal buffer area of the SLWA. The last record pointer is set to zero to indicate 'buffer empty'. The loop conter is set to 10.			

NOTES	MODULE	LABEL	REF
2A 3 The current position within the just processed member is saved, if a next nesting level will be entered. Later at reentry this position has to be reestablished. This is indicated by a flag in the SLME.	GETSL MTSL		



IPW$$AT - Abnormal VSE/POWER Termination Processing

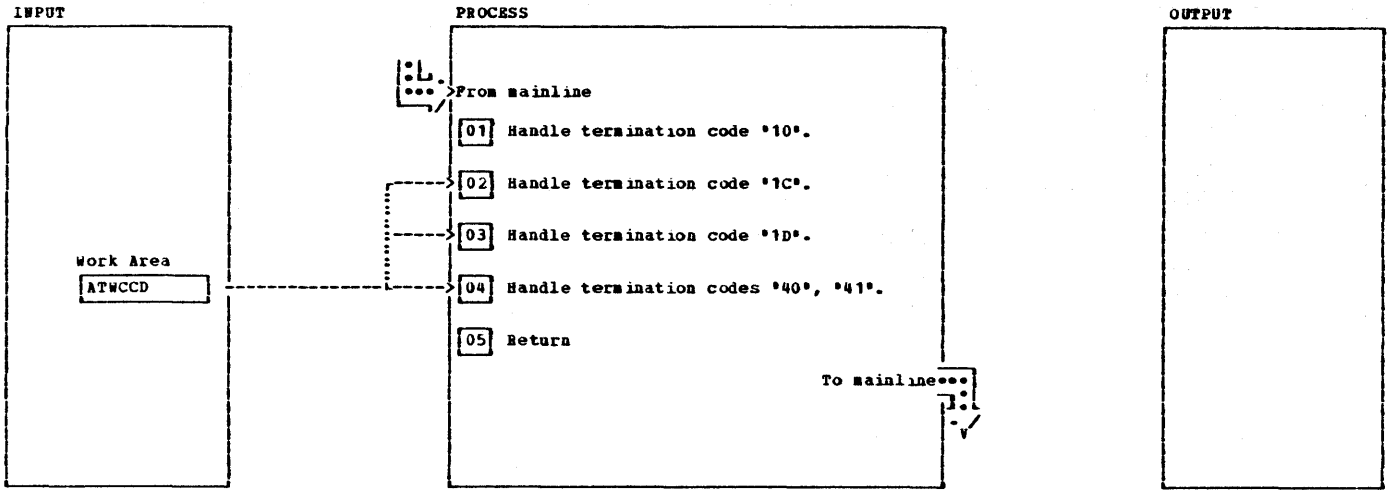


IPW$$AT - Abnormal VSE/POWER Termination Processing

NOTES	MODULE	LABEL	REF
**********			
DOS/VSE passes control to the AB-exit routine when an abnormal termination condition for the VSE/POWER maintask or one of its subtasks is encountered. Register 0 contains the cancel code (termination code) in its last byte. Register 1 contains the address of the abnormal register save area in which DOS/VSE places the registers at the point of termination.			
When VSE/POWER detects an error condition, which don't allow it to continue, it invokes the AB-exit directly by issuing the 'IPWCNC cancel' macro instruction. Register 0 contains the cancel codex 'FF' and the most current register contents reflecting the error condition are saved in the maintask abnormal register save area.			
**********			
1 The DOS/VSE STXIT macro instruction is issued in order to activate the linkage to the AB-exit routine.	STXIT	ATINIT	
Linkage to the AB-exit routine is established for all subtasks by specifying the ABSAVE parameter when attaching the subtask.			
The routine is invoked the first time at VSE/POWER initialization time.			
2 The workarea stack, which should be large enough to contain at least one entry for each subtask and VSE/POWER maintask is scanned to locate an unused slot. If found, the slot is reserved and the cancel code as well as the address of abnormal register save area are stored in it. If no free workarea slot is available, a program check is forced.			
3 All resources which are still owned by the VSE/POWER maintask or one of its subtasks are released.	UNLOCK		
4 Depending on the cancel code as provided by DOS/VSE, special actions are necessary.		AT050	
A message is prepared showing the PSW and the cancel code as well as an short description of the cause of the error. The message is then written to the central operator.			
102CI PSW = xxxxxxxxxxxxxxxx, CC = id cause			

NOTES	MODULE	LABEL	REF
5 The TIK of the VSE/POWER maintask, as obtained via the GETFLD macro is compared with the TIK currently in SYSCOM. If they are not identical, it is an indication that a VSE/POWER subtask cancelled.	GETFLD		
6 A channel program is built consisting of a write CCW to write message:	EXCP	AT300	
1030D ABNORMAL VSE/POWER TERMINATION, PRINTER= followed by a read CCW to get the operator's answer			
If the cancel key was hit, the message is re-issued.	WAIT		
6A The operator replied with a printer address. If the address is invalid message:		AT320	
1030D INVALID PRINTER TYPE, RE-ENTER			
is printed. If a valid address was given it is translated to a hexadecimal device address.			
6B The device address is then checked against the PUB table. If the device is not known or is down, or not a valid printer device (device type code not between X'40' and X'50'), message:		AT350	
1030D INVALID PRINTER TYPE, RE-ENTERED			
is printed.			
Otherwise the PUB index is stored in the SYSLST LUB entry and the dump option is set in COMREG			
7 The number of supported partitions is obtained from SYSCOM. Each partition COMREG is addressed by getting its address from the PIB2.	ASYSCOM	AT400	
The POWER flag in COMREG is then examined if the partition is running under control of VSE/POWER.			
9 If the operator wants a dump to be taken, the DUMP macro is issued, otherwise the CANCEL macro is issued.	CANCEL DUMP	AT600	

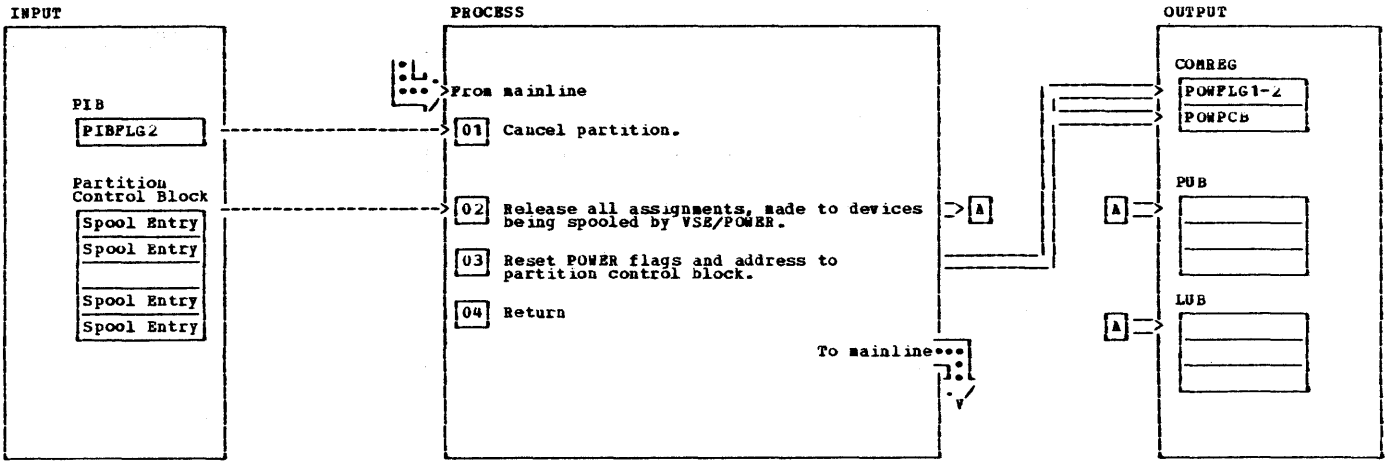
IP@$$AT - Handle Special Termination Codes



NOTES	MODULE	LABEL	REF
Depending on the termination code, passed either by DOS/VSE or VSE/POWER, special actions are necessary.			
1 This cancel code is given when when either the VSE/POWER maintask or one of its subtasks normally terminated. Immediate return to DOS/VSE is made.	EOJ		
2 This cancel code is given when a subtask issues a 'CANCEL ALL' macro. It causes to terminate all other subtasks as well as the VSE/POWER maintask to terminate with obove cancel code. When the AB-exit was invoked for a subtask, the subtask is detached.	DETACH	AT@1C	

NOTES	MODULE	LABEL	REF
3 This cancel code is given to all subtasks which are still alive, when VSE/POWER terminated either normally or abnormally. The subtask is detached.	DETACH	AT@1D	
4 This cancel code is given when VTAM terminates (cancels) the VSE/POWER subtask or when VTAM detects an invalid condition.		ATVTE	
A message showing the PSW and the cancel code is issued, and the immediate stop is posted in the SNA control block. The interface to VTAM is terminated by means of the Close ACB macro. Also the SNA manager is posted.			

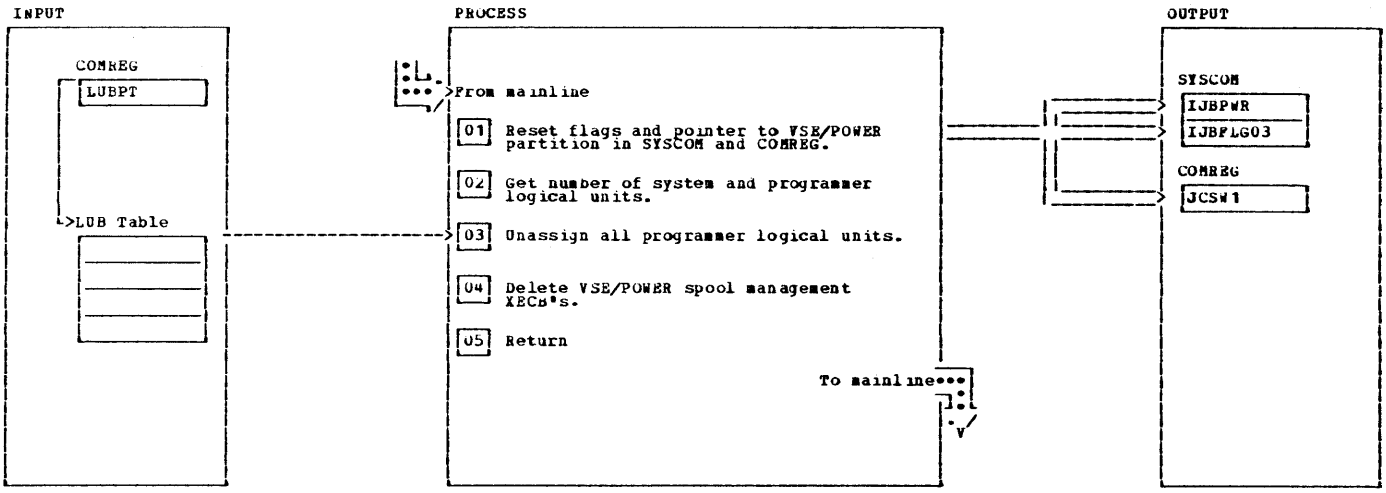
IPW$$AT - Cancel Partition Controlled by VSE/POWER



NOTES	MODULE	LABEL	REF
This routine cancels all partitions, controlled by VSE/POWER with cancel code X'23', and releases all UR-devices being spooled by VSE/POWER.			
1 If the partition is not already in stop state, it cancels with cancel code X'23'.	TREADY	CNCL00	

NOTES	MODULE	LABEL	REF
2 Each spool entry in the partition control block of the partition concerned is used to scan through the partition LUB-table to locate any entry which is assigned to a device spooled by VSE/POWER. If so, the assignment is reset, except for devices which are device type 'do not intercept'.	MSAT	CNCL20	

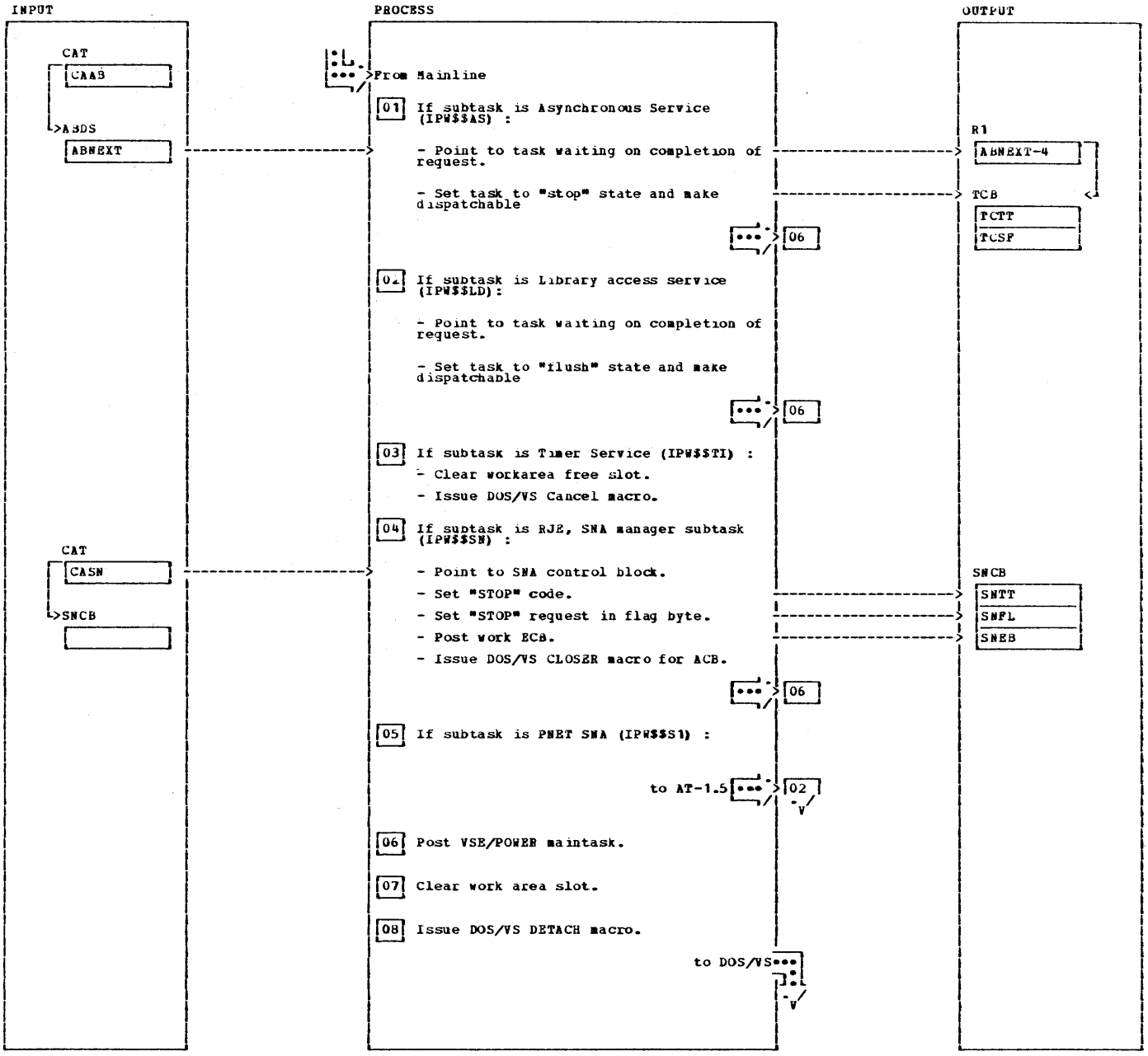
IPW$SAT - Clean Up VSE/POWER Partition



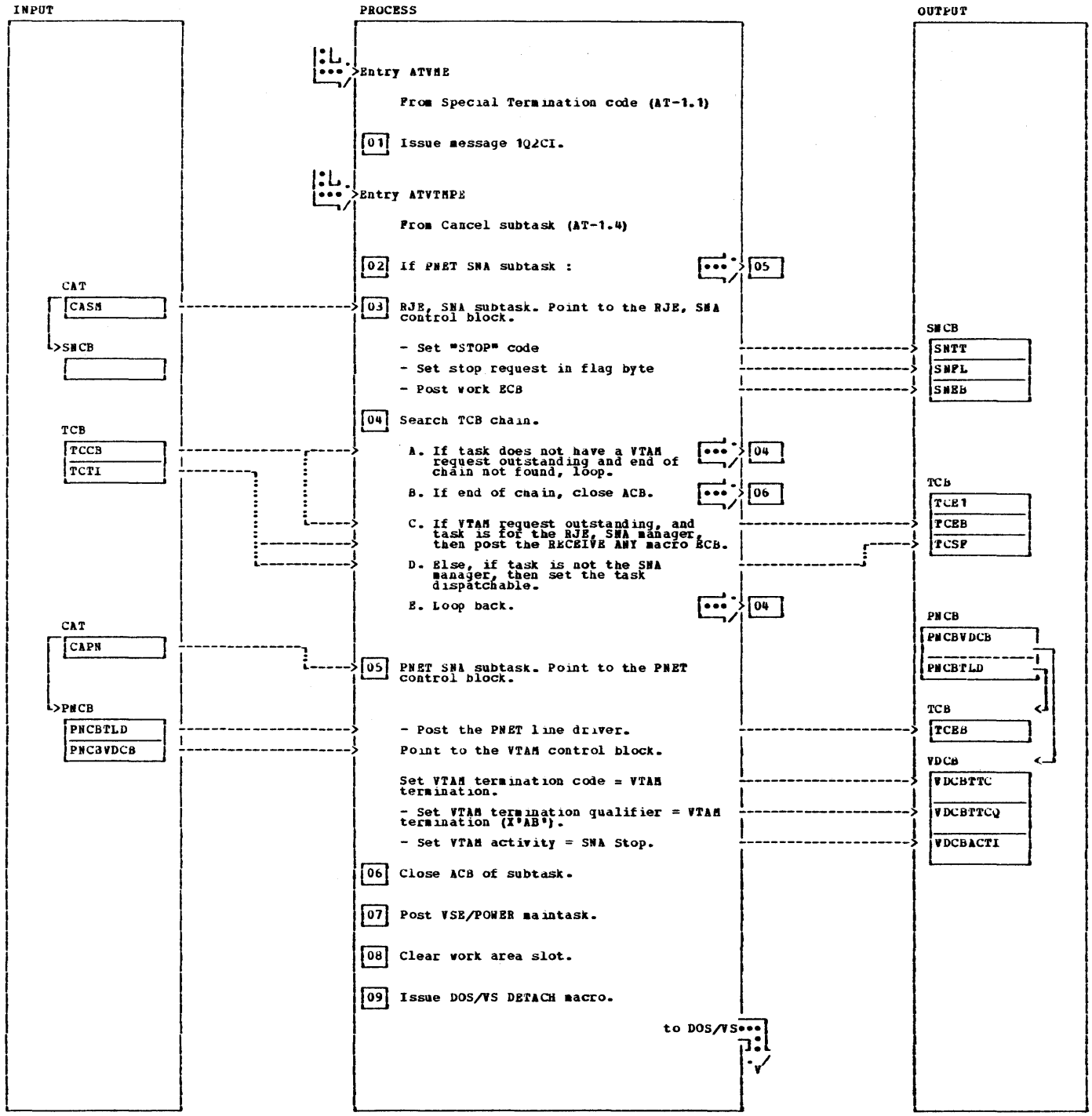
NOTES	MODULE	LABEL	REF
1 The VSE/POWER active flag in SYSCOM is reset and the address of the VSE/POWER partition is deleted.	ASYSCOM	AT500	
Reset the job control switch for the partition COMREG.	COMRG		
3 The VSE/POWER LUB table is scanned, starting with the first programmer logical unit, and each logical unit which is not assigned to the disk device, is unassigned and the PUB ownership cleared.	MSAT		
If the device is a 3800 printer, a SETPRT parameter list is built and a SETPRT request is issued, which sets up the printer with system/hardware defaults.		AT550	

NOTES	MODULE	LABEL	REF
4 The macro IECBTAB is used to first delete the internal reader IECB and next the spool manager IECB.	IECBTAB		
Then the IECBTAB macro is used to check if the internal reader is active. If so, the XPOST macro is used to notify the user that VSE/POWER is terminating.			
A check is made if the spool manager interface is active. If so, the user is notified that VSE/POWER is terminating.	XPOST IECBTAB		

IPW$$AT - Cancel Subtask



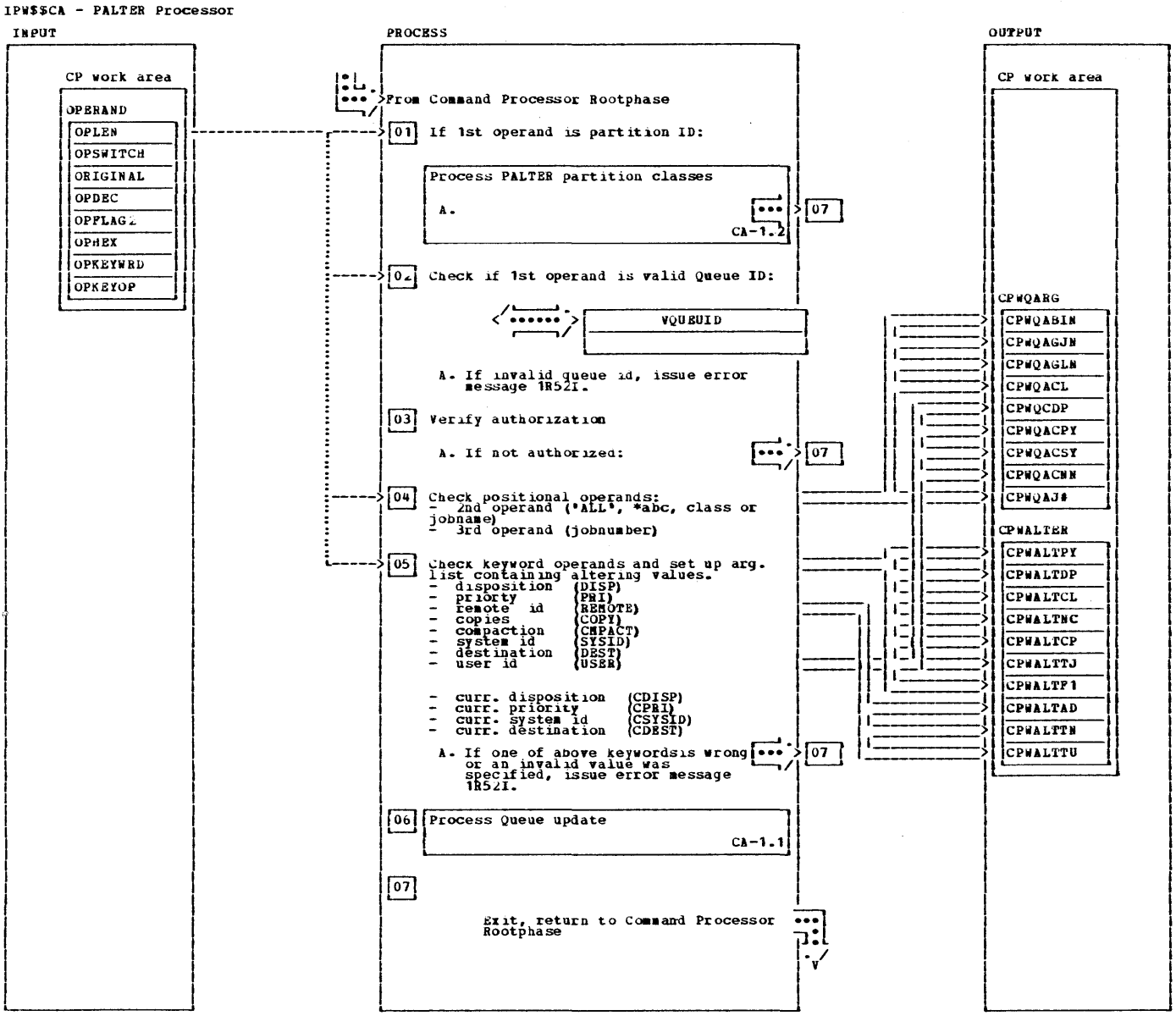
IPW$$AT - VTAM Termination Handling



IPW$$AT - VTAM Termination Handling

NOTES	MODULE	LABEL	REF
This routine is entered at the label ATVTM when the subtask has been canceled due to VTAM termination or invalid VTAM condition (codes x'40' and x'41' respectively). The entry is at ATVTMPE when the PNET SNA subtask has been canceled.			
1 The following message is issued IQ2CI PSW= xxxxxxxxxxxxxxxx CC=id VTAM Termination	ATVTM		
2	ATVTMPE		
4 Reactivate any RJE, SNA tasks that are waiting on completion by VTAM execution macros (the bit TCVR is set in the TCB). This causes the task to run to the next VTAM macro (in most cases the CHECK macro) which will give an error return code (X'20') due to the VTAM interface having been closed (in following process step). The task will then enter its own termination code, and detach.	ATVTM5		
6	CLOSER		
7	TREADY		
9	DETACH		

CHART CA: IPW$$CA

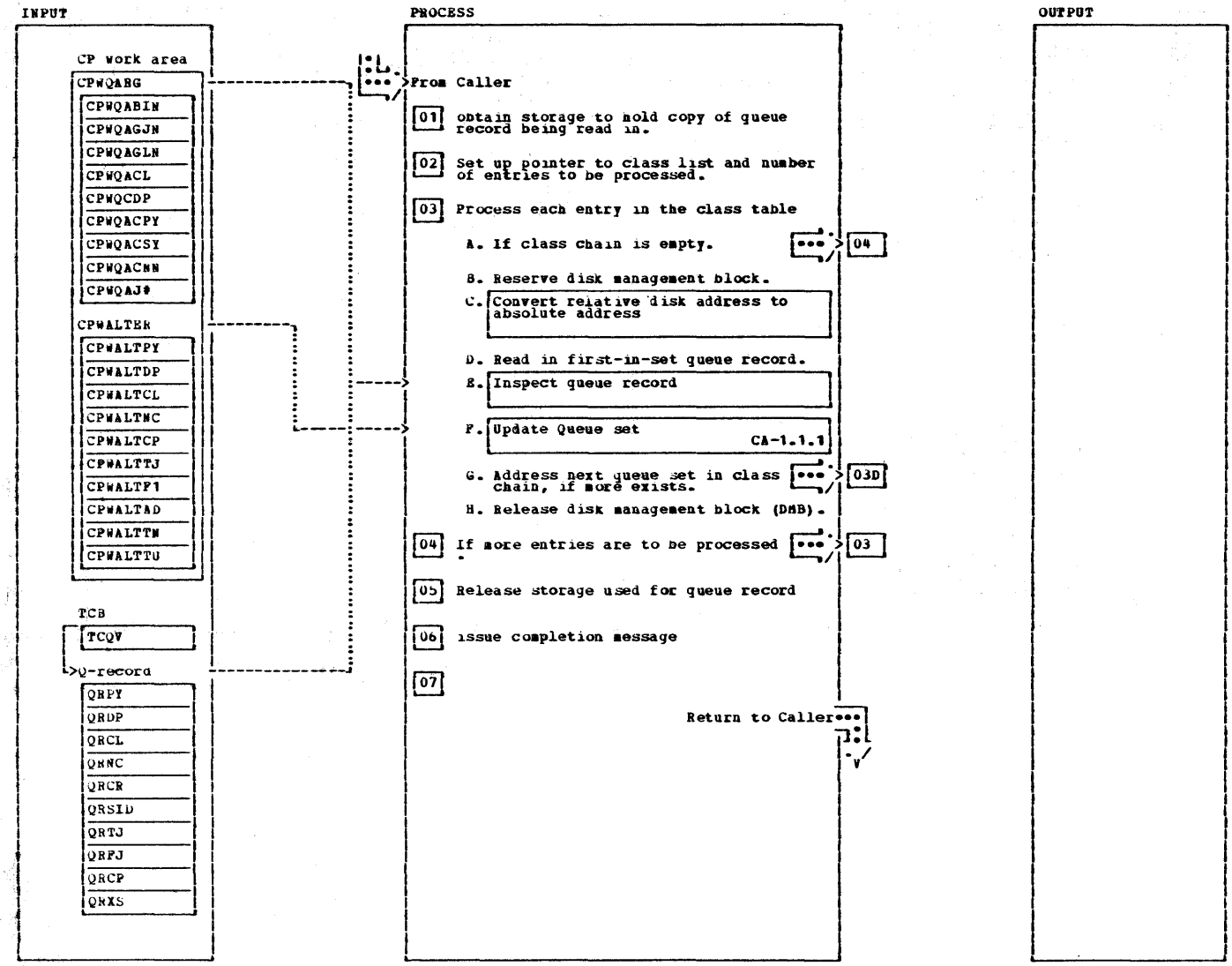


IPW$$CA - PALTER Processor

NOTES	MODULE	LABEL	REF
The PALTER command has one of the following formats: PALTER queue,jobname,<jobnumber> keyword=value PALTER queue,ALL, keyword=value PALTER queue,class, keyword=value PALTER queue,*abc, keyword=value PALTER partition<,class(es)> where: queue : One of LST,RDR,PUNor XMT jobname : A jobname known to VSE/POWER jobnumber: A jobnumber assigned to jobname ALL : Specifies that all jobs in queue are to be altered class : Specifies that all jobs in queue for this class are to be altered *abc : Specifies that all jobs in the queue whose names have the specified characters in common are to be altered (generic jobname) keyword : One of PRI,DISP,CLASS,COPY, REMOTE,CMPACT,SYSID, DEST,USER,CDISP, CDEST,CSYSID,CPRI value : Specifies the new value for an attribute partition: Specifies a VSE/POWER controlled partition for which classes shall be altered class(es): Specifies the new input classes of the jobs which may be executed in that partition		PALT000	

NOTES	MODULE	LABEL	REF
1 The 'Verify partition id' subroutine is called to check if the first operand is a valid partition id. the subroutine returns in field 'QTHPIBPP' the address of the partition concerned or the field contains zero if not a valid partition.			
2 The 'Verify Queue id' subroutine is called to examine if the operand specifies a valid queue id. Valid queue ids are - RDR, LST, PUN or XMT.			
2A Message 1R52I ccccccccc OPERAND # # MISSING OR INVALID is issued.			$GAM
3 The 'Verify command authorization' subroutine is called to check if the issuer of the command is authorized. If not, the command is rejected. An appropriate error message has been already issued by the subroutine.			$VCA
4 If invalid specifications have been entered message 1R52I ccccccccc OPERAND ## MISSING OR INVALID will be issued.			$GAM
5 Several keywords with their associated values may be entered by one PALTER command		PALT400	
5A One of the following messages will be issued: 1R52I ccccccccc NO VALID KEYWORD SPECIFIED 1R52I ccccccccc OPERAND ## NOT SPECIFIED AS KEYWORD 1R52I ccccccccc INVALID SPECIFICATION FOR KEYWORD			$GAM

IPW$$CA - Process Queue update

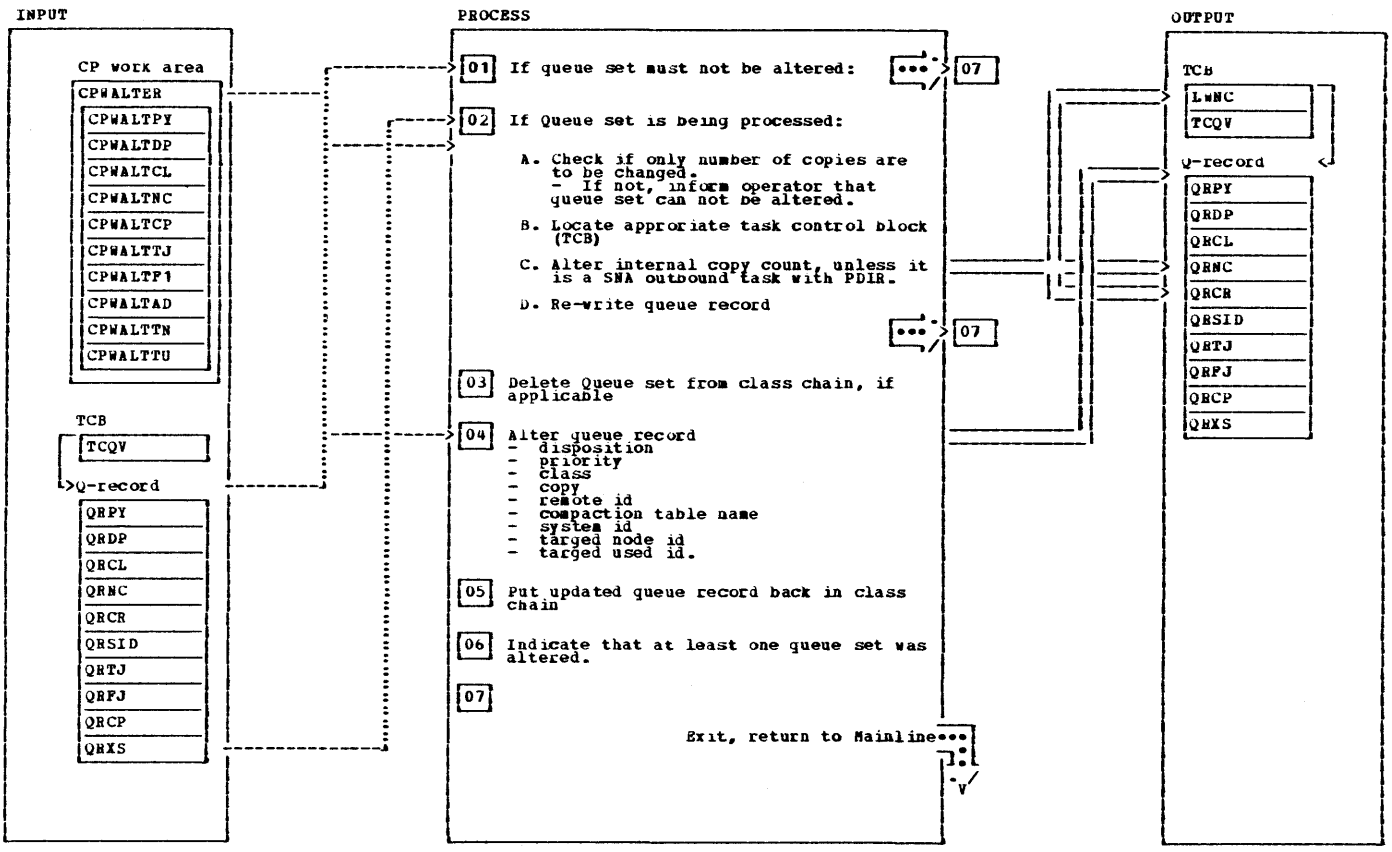


IPW$$CA - Process Queue update

NOTES	MODULE	LABEL	REF
The reader , list , punch and/or xmit queue is scanned and each queue set which is eligible is extracted and updated.			
1 For the queue record area workspace will be provided.		PALT500	$RSW
3B The disk management block is reserved for the duration of the queue scanning.			$RSB
3C Subroutine RELTOABS converts the relative disk address			
3D The first in set queue record is read in the queue record area just acquired.			$RDQ
3E The queue record just read in is examined if it is eligible to be updated. - jobname and number - class - RJE user id - local or remote destination - target node name - curr. disposition - curr. priority - curr. destination.			
3F A queue set can be altered if it is not being processed unless only the number of copies is to be changed. If this is not true message 1R88I cccccccc JOB jjjjjjjj nnnnn CANNOT BE ALTERED will be issued. When copy and/or priority attributes may have to be altered the queue set is deleted from queue chain. After doing the appropriate changes the 1st-in-set queue record is written back. In case of changing class and/or priority attributes the queue set is added back to the class chain according to its priority.			

NOTES	MODULE	LABEL	REF
3G Queue sets belonging to a class are chained via the 'QCQM' field. This field contains the absolute disk address of the next queue set in chain. For the last queue set, this field contains binary zeros.			
3H The disk management block is released.			$RLB
6 If any queue set is altered message 1R88I OK for the central operator only or if not 1R88I NOTHING TO ALTER will be issued.			$GAM

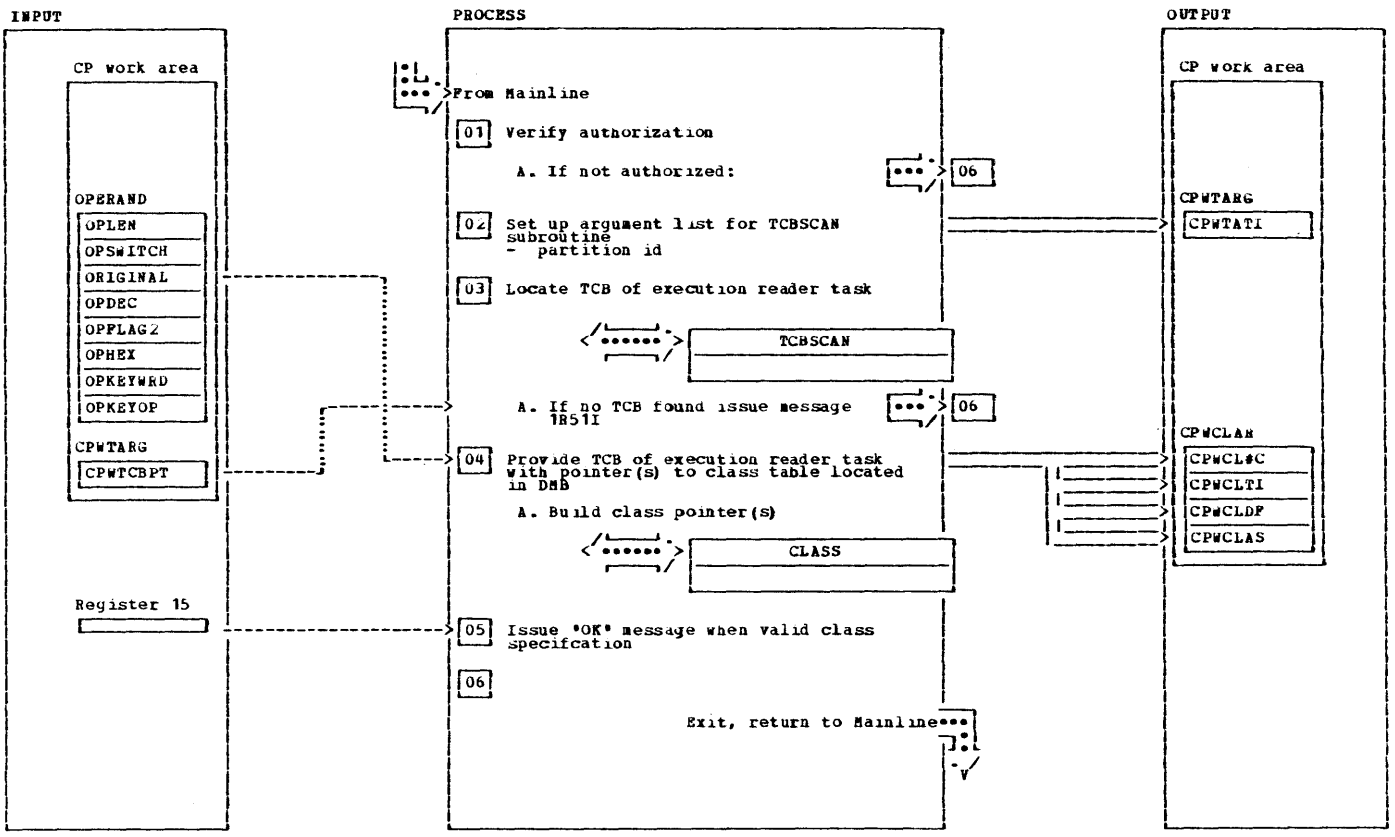
IPW$$CA - Alter Queue set



NOTES	MODULE	LABEL	REF
1 If the result of the PALTER process would be that the attributes of the queue set, just being examined, remain as they were, nothing will be done.		PAQR100	
2 A queue set can be altered if it is not being processed unless only the number of copies is to be changed. If this is not true message 1R881 cccccccc JOB jjjjjjjj nnnnn CANNOT BE ALTERED will be issued.			$GAM

NOTES	MODULE	LABEL	REF
2D			$WTQ
3 When number of copies, class, priority and or destination are subject of the PALTER command, the queue set is removed form the class chain.			$DQS
5 When the queue set was daleted from the class chain prior it is now added back to the class chain according to its priority.			$AQS
5 In the other case, the first-in-set queue record is just written back.			$WTQ

PALTER - Partition classes

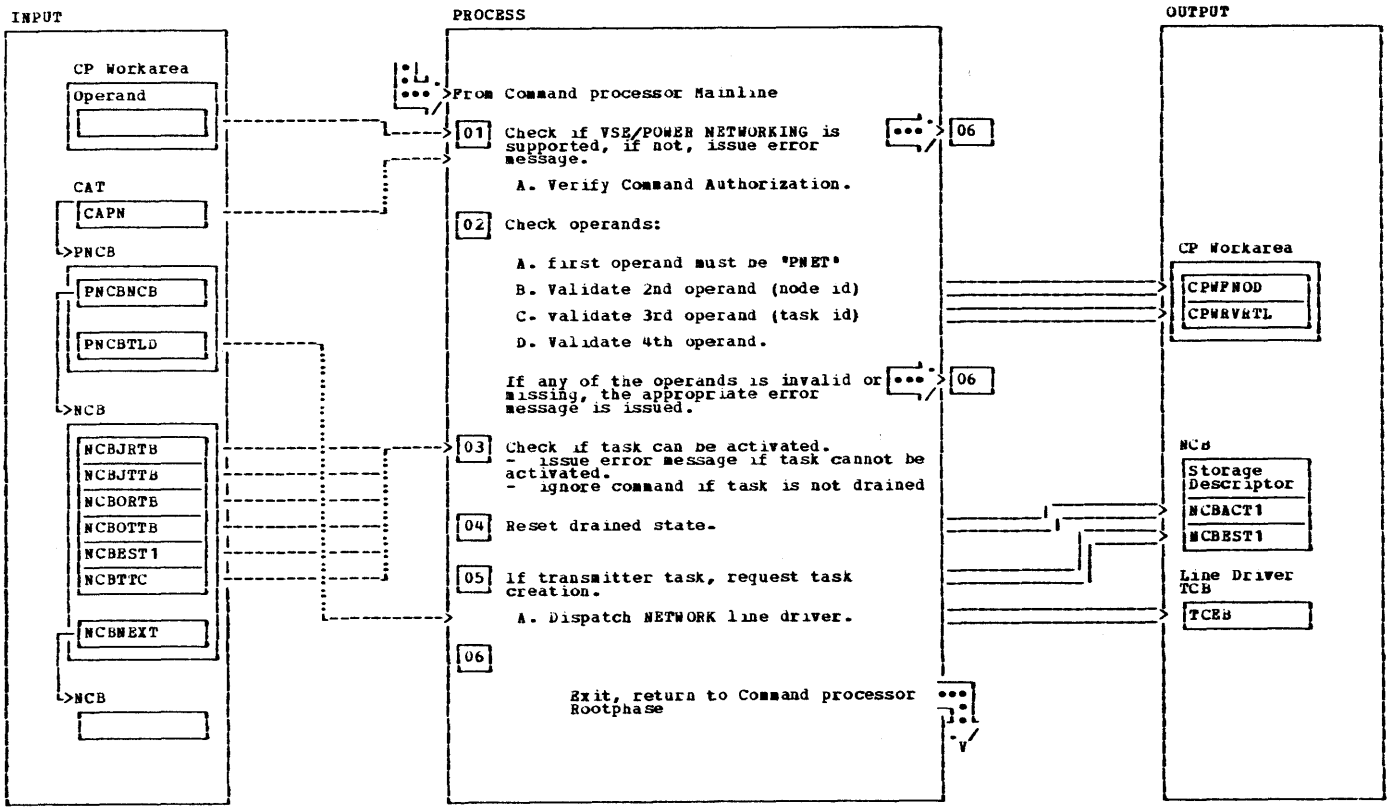


NOTES	MODULE	LABEL	REF
1 The 'Verify command authorization' subroutine is called to check if the issuer of the command is authorized. If not, the command is rejected. An appropriate error message has been already issued by the subroutine.			$VCA
3 Task identifier is 'E xx' where xx is partition id			

NOTES	MODULE	LABEL	REF
3A Following message is issued: 1R51I cccccccc OPERAND ## DESIGNATES NON-EXISTING TASK			
5 In case of invalid class specification subroutine CLASS has already issued error messages, so a branch to processor exit is taken.		PAPC210	
5 Following message will be issued to central operator: 1R88I OK			$GAM

This page was left blank intentionally.

IPW$CAC - PACT Processing

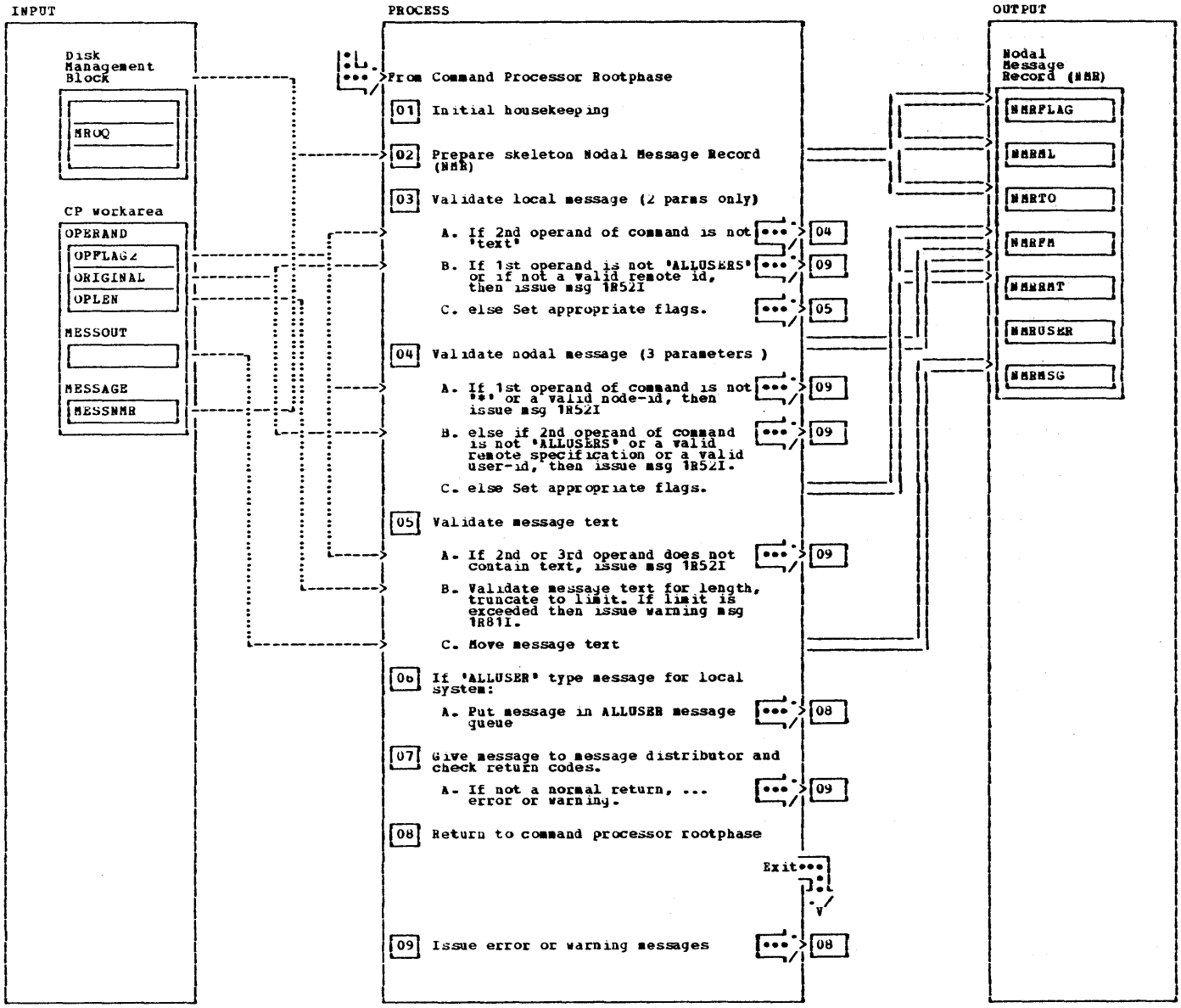


NOTES	MODULE	LABEL	REF
1 A first a check is made if VSE/POWER networking is supported. If not, following error message is issued: 1RA3I VSE/POWER NETWORKING NOT SUPPORTED A 2nd check verifies the Authorization of the issuer.		PACTN00	
2A If the first operand is not 'PNET' following error message is issued: 1R52I OPERAND ## MISSING OR INVALID			
2B The second operand specifies the node id for the task to be activated. The node id must be alphameric, first character alphabetic and up to a max. length of eigth bytes. If the operand is errornous, message 1R52I OPERAND ## MISSING OR INVALID is issued.			
2B The NCB chain is scanned to locate the appropriate NCB. If no NCB is found, indicating that no connection is established or even that the node is unknown, message 1RB1I NO CONNECTION ESTABLISHED TO NODE nnnnnnnn is issued.			
2C The 3rd operand is mandatory and specifies the task id of the task to be activated. Valid specifications are 'TR1 - TR7' for transmitter tasks or 'RV1 - RV7' for reveiver tasks. If no valid task id is specified, message 1R52I OPERAND ## MISSING OR INVALID is issued.			

NOTES	MODULE	LABEL	REF
2D The 4th operand specifies what the task to be activated is supposed to do, either to process jobs or output (LST or PUN). Valid specification is 'OUT' or 'JOB'. If none of above specifications, message 1R52I OPERAND ## MISSING OR INVALID is issued.			
3 The command rootphase has already verified that VSE/POWER is not in shutdown period. Now the termination code in the NCB is examined if the connection to the specified node is in stop state. If so, following Message will be issued: 1RC3I cccccccc COMMAND REJECTED, nnnnnnnn IN SHUTDOWN PERIOD. The command is ignored when the task going to be activated is not in drained state. Only seven transmitters of any sort can be activated conrurrently. The total number of transmitters is limited to eigth, so certain job transmitters exclude corresponding output transmitters. (e.g. TR3 OUT excludes TR6 JOB). In that case following Error Message will be issued: 1RC2I cccccccc TRANSMITTER CANNOT BE STARTED.		PACTN60	
4 The drained flag is turned off in the appropriate task table contained in the NCB.			
5 The task creation flag is turned on in the appropriate NCB and the Line Driver ECB is posted.			

This page was left blank intentionally.

IPW$$CB - PBRDCST - Processor



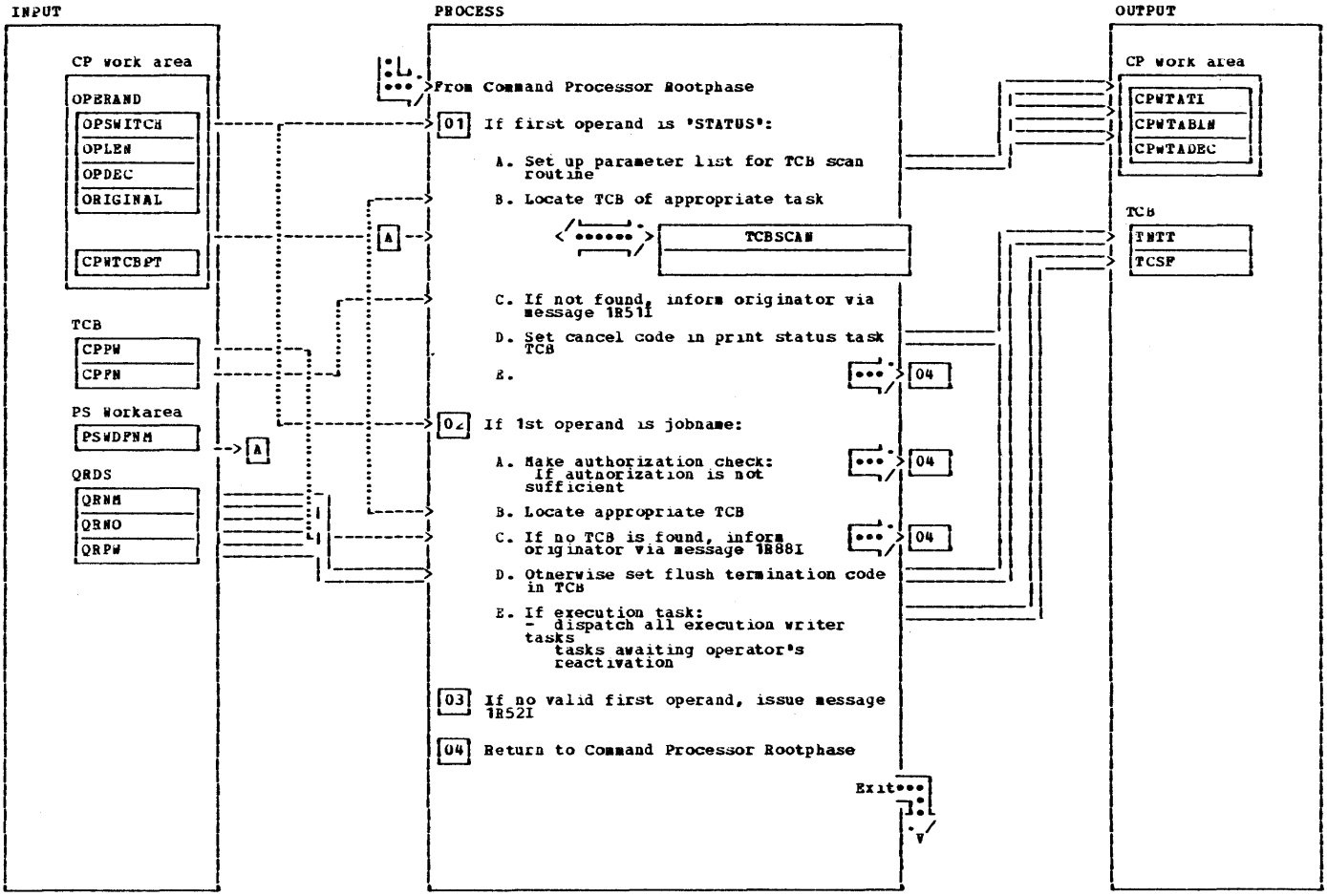
IPW$$CB - PBRDCST - Processor

	NOTES	MODULE	LABEL	REF
1	The Command processor rootphase registers are stored in the save area,reserved for this purpose. Addressability to various control blocks is set up.		PBCST100	
2	The skeleton NMR Record is set to defaults: NMRTO and NMRFM contain own Node name and SYSID. NMRSL contains the maximum message length.			
3	The message text has to be in single quotes, i.e. byte OPSUQUO in OPFLAG2 is on. If text is available,the first operand can only be 'ALLUSERS' , a one or three digit numeric remote-id, a remote id in 'Rnnn' format or '*', in which case the message is destined for local operator. Remote-id = 0 means: Central operator. Else issue message: 1R52I command code OPERAND ## MISSING OR INVALID. If it is an 'ALLUSERS'-type message,the max.length is set to 46.		PBCST200	
4	The node-id has to be * ,i.e local node or 1-8 alphameric. The user-id can be 'ALLUSERS' or correct remote-id as described above,or 1-8 alphameric user-id (from another node-id). If not issue error message 1R52I		PBCST300	

	NOTES	MODULE	LABEL	REF
5	If it is a local message the 2nd operand has to be 'text',otherwise the 3rd operand has to be 'text'. If not issue error message 1R52I.... Compare the length of the text with the max.length and if exceeds truncate message and issue warning: 1R81I command code MESSAGE TEXT WILL BE TRUNCATED.		PBCST400	
6	The remote message service is called to put the ALLUSER type message in the queue. If an error condition occurred, message 1R92I command code ALLUSERS QUEUE IS FULL is issued.			$RMS $GAM
7	If it is an unnormal return the following messages will be issued: 1R65I command code RJE NOT SUPPORTED 1RA3I command code VSE/POWER NETWORKING NOT SUPPORTED 1R92I command code ALLUSERS QUEUE IS FULL 1R93I command code REMOTE CURRENTLY NOT SIGNED ON 1R65I command code INVALID REMOTE ID 107AI command code NO VIRTUAL STORAGE AVAILABLE 1RB3I command code NO CONNECTION ESTABLISHED TO NODE nnnnnnnn		PBCST500	$gms
9	All the messages described above will be issued.		PBCST700	$gam

This page was left blank intentionally.

IPW$$CC - PCANCEL Processor

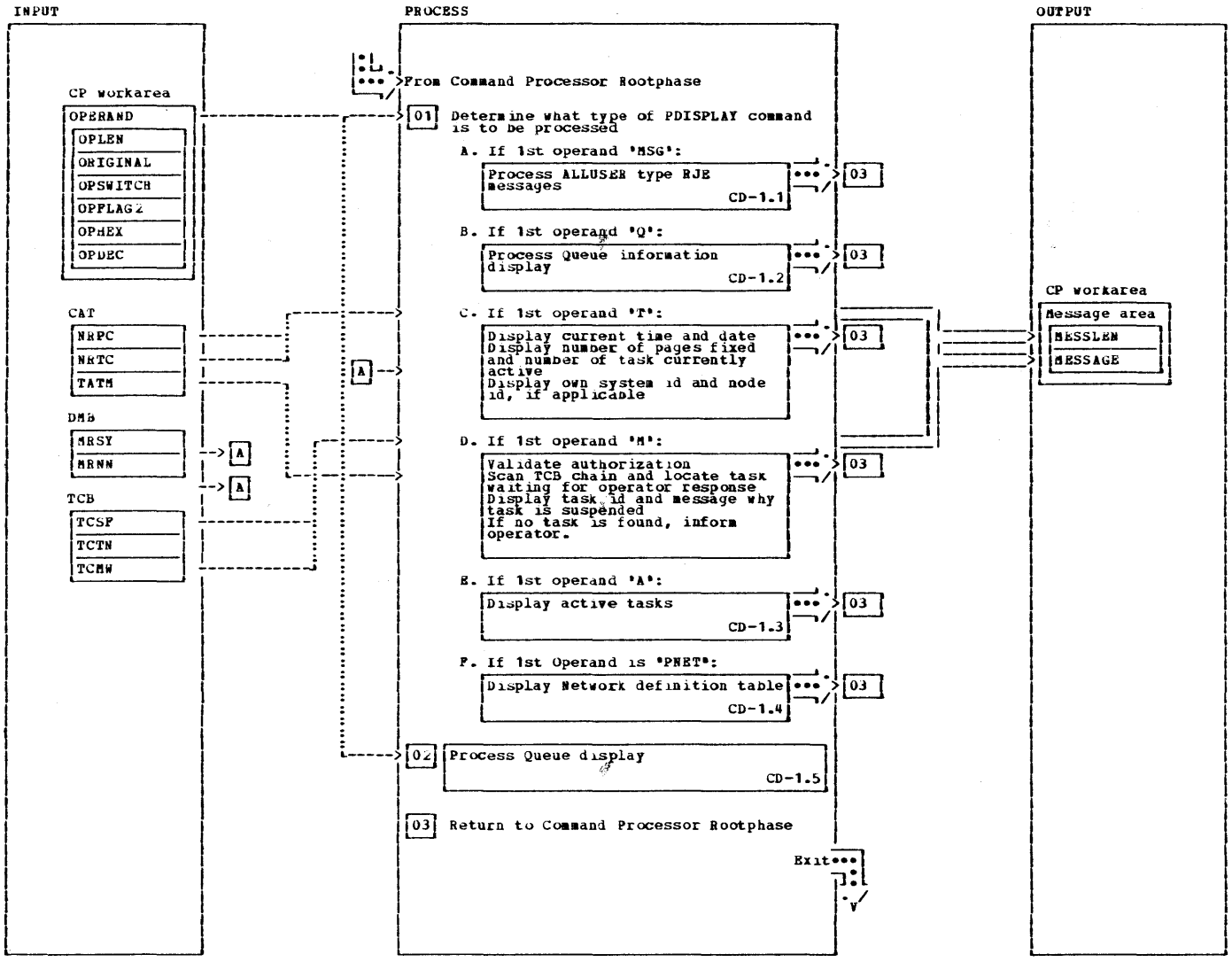


IPW$$CC - PCANCEL Processor

	NOTES	MODULE	LABEL	REF
	The PCANCEL command has one of the following formats: PCANCEL <STATUS> or PCANCEL jobname<,jobnumber> where: jobname : A jobname known to VSE/POWER jobnumber : A jobnumber associated with the jobname		PCNL00	
1A	Task identifier for print status TCB is 'P PS'.		PCNL10	
1B	The 'TCB scan' routine, is called to locate the TCB meeting the arguments setup by the caller.			
1B	If the command arrived over the network (originated by another node), the from node name in the command control block is checked against the one in the PS workarea. If the node names do not match, which means a print status TCB was located serving a different PDISPLAY, the TCB scan is continued.			
1C	Following message will be issued: 1R51I command code NO STATUS REPORT IN PROGRESS			$GAM
2B	The TCB chain is scanned to locate either execution reader or physical reader/list or punch task. If such a task TCB if found the queue record anchored in the TCB of the appropriate task is examined whether it is the queue entry (jobname) supposed to be canceled If the PCANCEL command was originated by a remote operator, the job supposed to be canceled must also be submitted by him. If the command is received from another node, that node is only allowed to cancel jobs (queue entries) which are originated from that node. In case a remote operator or user of another node, issued the PCANCEL command, the originating user id must also match.			

	NOTES	MODULE	LABEL	REF
2C	Following message will be issued: 1R88I NOTHING TO CANCEL			$GAM
2E	All spool entries in the partition control block associated with the execution reader task are examined if a execution writer task is waiting for re-activation by the operator. If so, stop code 'S' is set in the execution writer task TCB and the task is set dispatchable.			
3	Following message will be issued: 1R52I cccccccc OPERAND ## MISSING OR INVALID			$GAM

IPW$$CD - PDISPLAY Processor

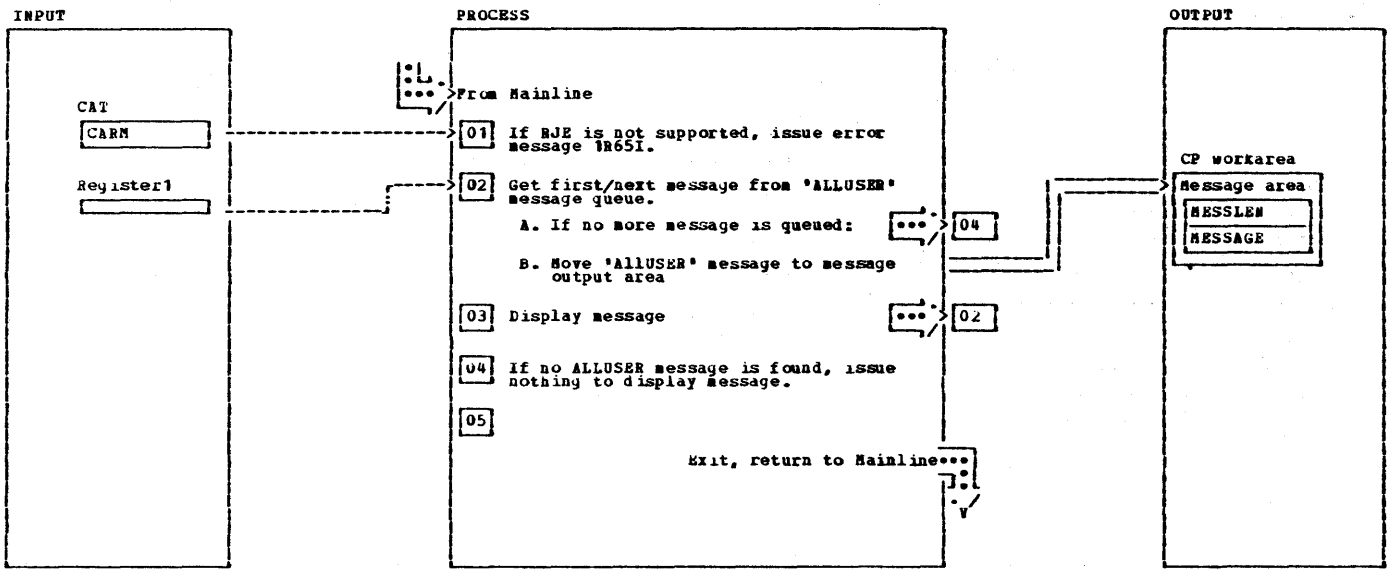


IPW$$CD - PDISPLAY Processor

NOTES	MODULE	LABEL	REF
The PDISPLAY command has the following format either: PDISPLAY queue,<,jobname><,jobnumber> <,keyword=value> PDISPLAY queue<,ALL> <,keyword=value> PDISPLAY queue,HOLD or PDISPLAY queue,FREE or PDISPLAY queue,RJE<,remid> or PDISPLAY queue,LOCAL or PDISPLAY queue,*abc or PDISPLAY queue,class or PDISPLAY XMT,PNET,node-id PDISPLAY ALL<,listaddr> or PDISPLAY HOLD or PDISPLAY FREE or PDISPLAY RJE<,remid> or PDISPLAY LOCAL or PDISPLAY *abc or PDISPLAY MSG or PDISPLAY A or PDISPLAY M or PDISPLAY Q or PDISPLAY T PDISPLAY PNET<,nodeid> or PDISPLAY PNET,LINKS or PDISPLAY PNET,ALL<,cuu> where is: queue : Specifies a LST, PRT, PUN, RDR or XMT queue jobname : A jobname known to VSE/POWER jobnumber: A jobnumber assigned to jobname keyword : One of CPRI, CDEST CDISP or CSYSID value : specifies current value of the attribute remid : A remote identifier abc : Specifies that all jobs in the queue whose names have the specified characters in common are to be displayed (generic jobname) class : Specifies that all jobs in the queue for this class are to be displayed node-id : specifies that all queue entries destined for that node are to be displayed. listaddr : Specifies a printer address in the form cuu or x'cuu' nodeid : A specific Node Name or if omitted Own entry from NDT cuu : specifies a printer address			

NOTES	MODULE	LABEL	REF
1 Depending on the first operand the determination is made, wether the queue status, current time, active task(s), task(s) waiting for operator response, pending messages, all user type RJE messages, network definition table information or job status is to be displayed.		PDSPL100	
1B The number of free queue records and the free space in the Account file, if applicable, are displayed.			
1C Following messages will be issued: 1R46I TIME IS xx:xx:xx:, DATE IS xx/xx/xx and 1R46I xxx PAGES FIXED, xxx CURRENT TASKS and optionally: 1R46I SYSID=x,NODEID=nnnnnnnn			$GAM
1D Each TCB in the task selection list is examined if it is waiting for operator response. If so, a message, containing the task id and the device address followed by the first 44-bytes of the message which has been issued just before the task entered the wait state, is prepared and displayed. If no task is waiting for operator reactivation, following message will be issued: 1R47I ccccccccc NO MESSAGE PENDING			$GAM
1E A list of all active reader and writer tasks, execution tasks and RJE tasks together with the name, number, and class of the job that is currently being processed is displayed.			

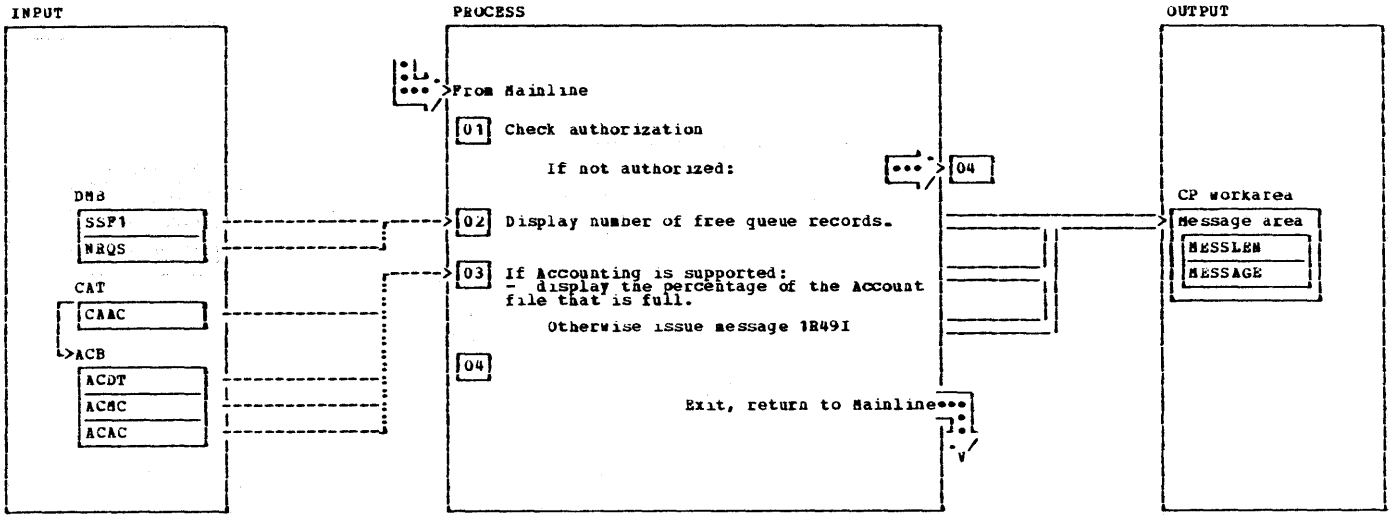
Process ALLUSER type RJE messages



NOTES	MODULE	LABEL	REF
1 2 Remote message service is invoked to return the address of the first message in register 1. Register 1 contains zeroes on 1st entry of remote message service. On exit register 1 points to the next message in message queue to be displayed		PDALMSG	$RMS

NOTES	MODULE	LABEL	REF
4 Following message will be issued: 1R46I cccccccc NOTHING TO DISPLAY			$GAM

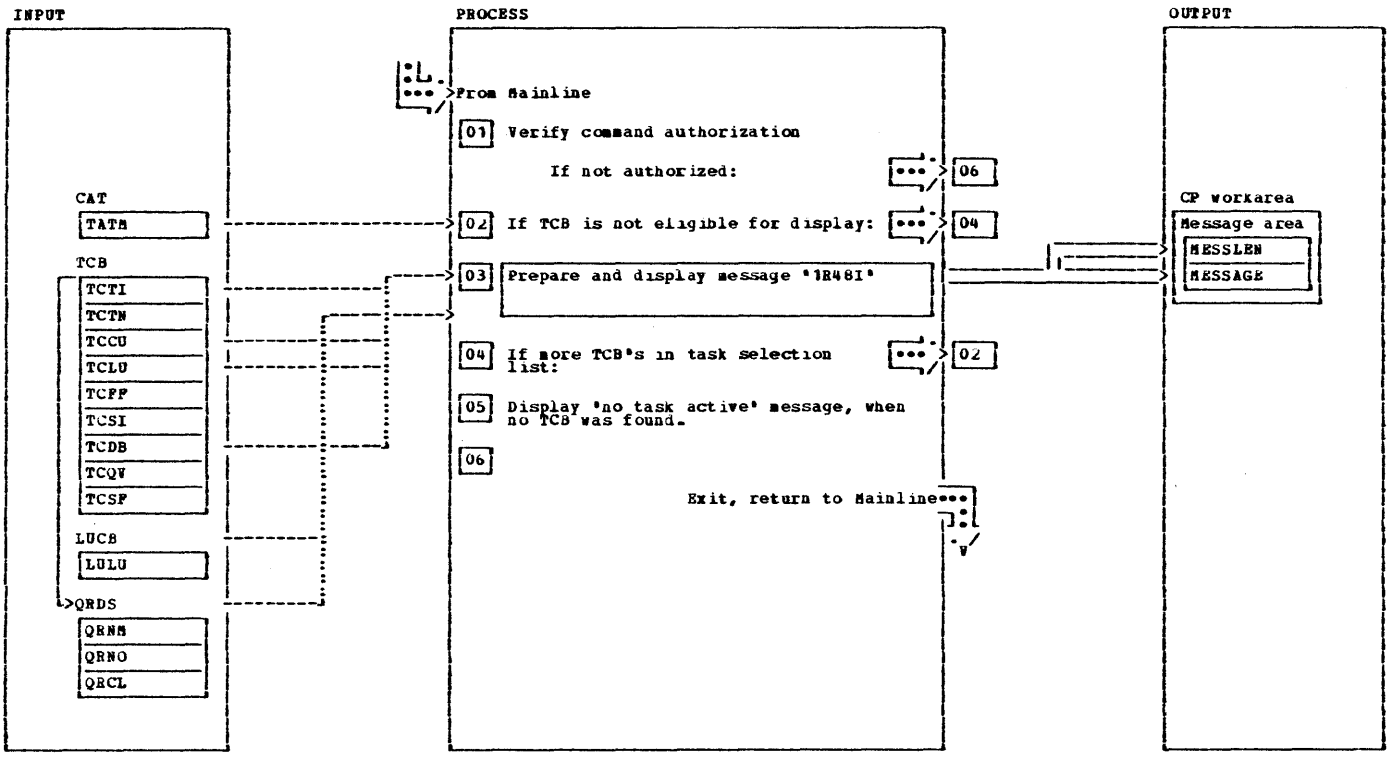
Process Queue info display



NOTES	MODULE	LABEL	REF
1 The command authorization subroutine is called to check if the issuer of the PDISPLAY command is authorized or not. If not, the command is rejected. The subroutine has already put out an error message, therefore an immediate branch is made to the processor exit.		PDFQR10	$VCA
2 If shared spooling, the master record is re-freshed to get the most accurate values. This is done by reserving and releasing the DMB again.			$RSR

NOTES	MODULE	LABEL	REF
4 The number of free records in queue fileis obtained and converted into printable format			$RLR
2 Following message is issued: 1R49I FREE RECORDS QFILE xxxxx			$GAM
3 'ACMC' in the ACB gives the maximum capacity of the account file. 'ACAC' gives the space available in the account file . Following message is issued: 1R49I ACCOUNT FILE xxx% FULL.			$GAM
3 Following message will be issued: 1R49I cccccccc NO ACCOUNTING SUPPORT			

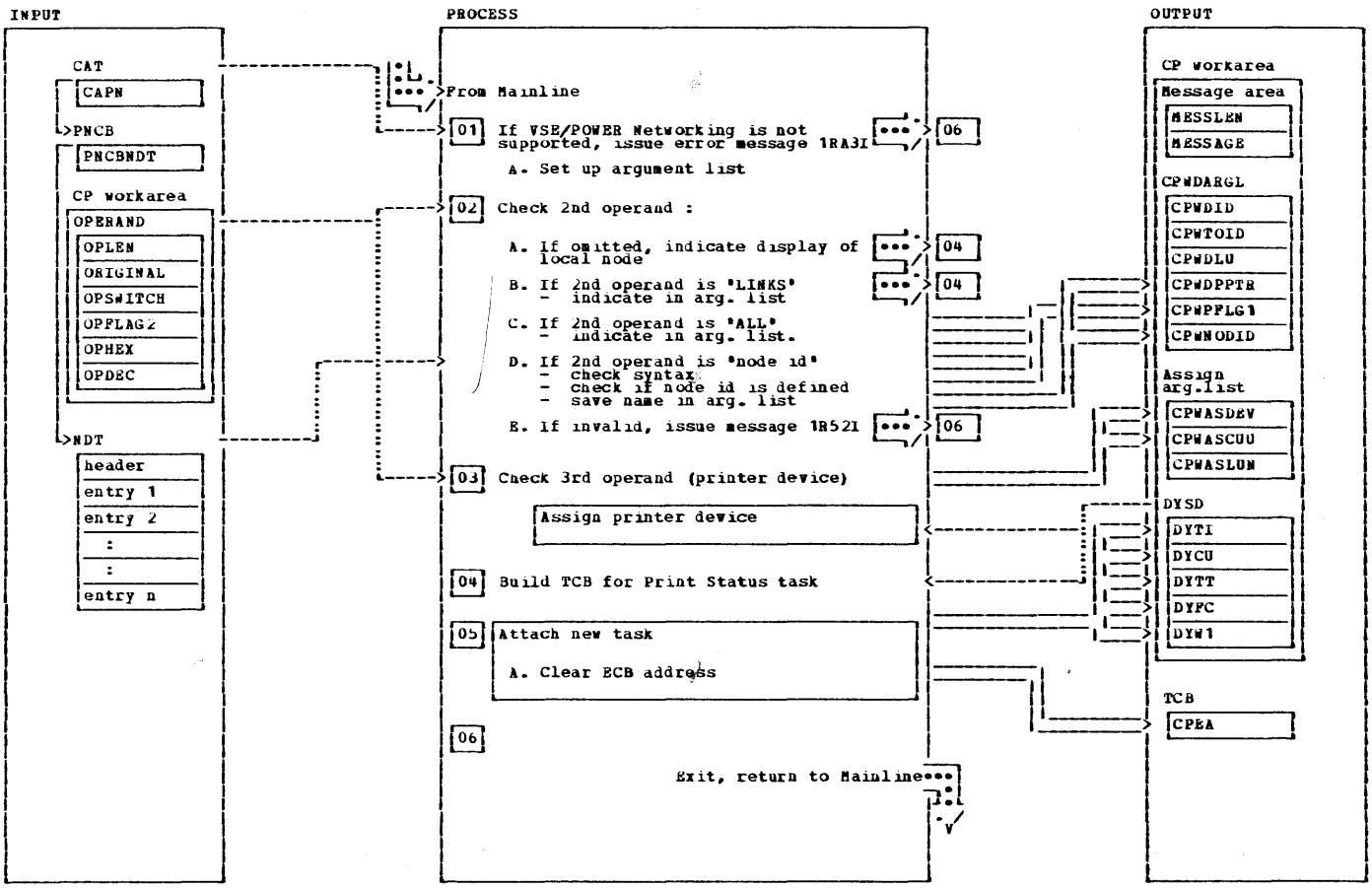
Process display of active tasks



NOTES	MODULE	LABEL	REF
1 The 'Command authorization' subroutine is called to check if the issuer of the command is authorized. If not, the routine has already issued an appropriate error message and the command is ignored .			$VCA
2 Each TCB in turn is examined if it is one of the following tasks: - execution tasks - physical reader/list/puncn tasks - RJE reader/writer tasks - RJE message tasks If so, the task is eligible for display.			

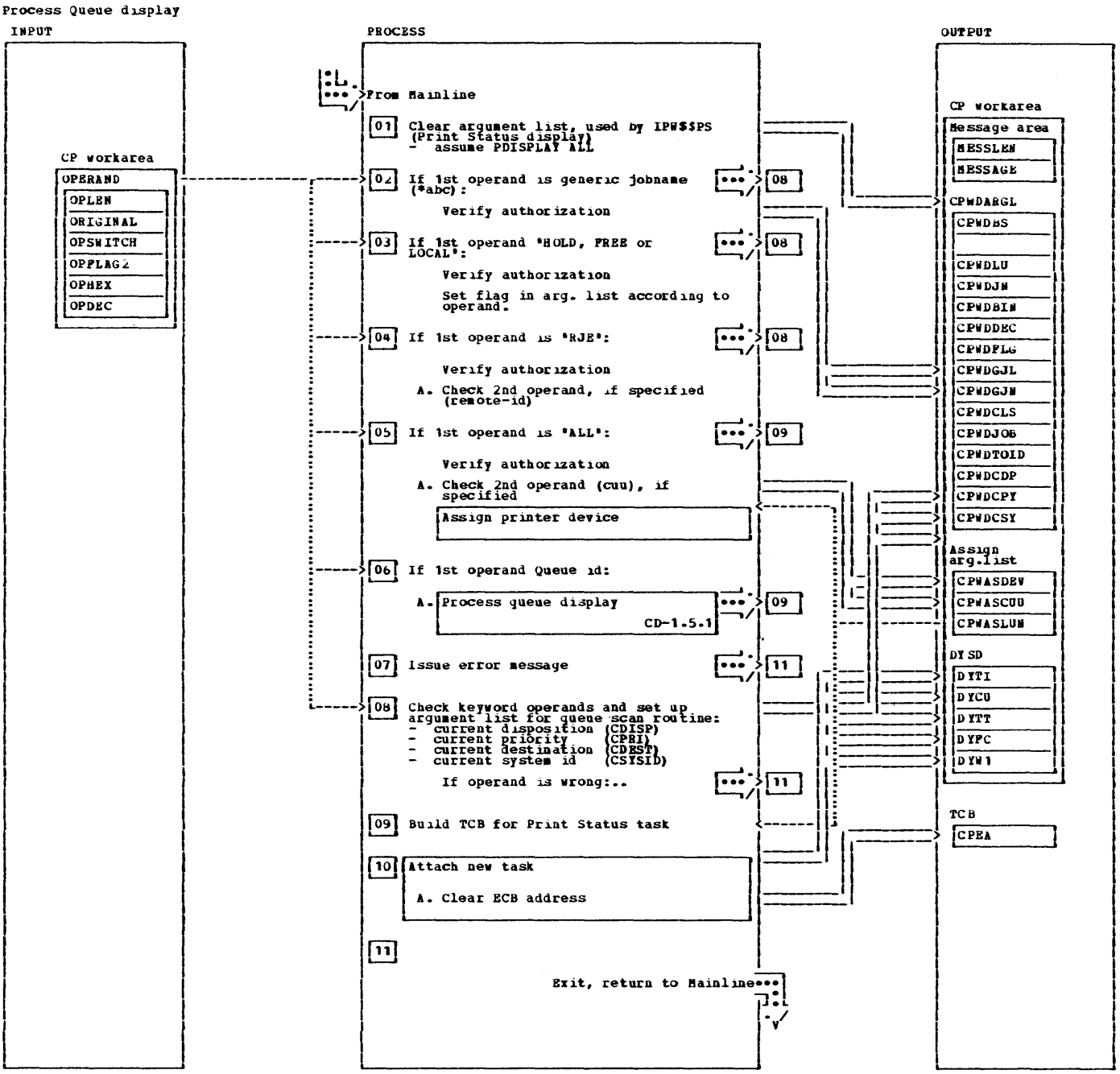
NOTES	MODULE	LABEL	REF
3 Following infromation extracted from the task control block (TCB) are displayed: - task id - device address being used - tape address if tape spooling - class - # of output buffers used - jobname and number - job class - 'INACTIVE' if task is waiting for work - logical unit name (RJE,SNA only)			
5 Following message will be issued: 1R48I cccccccc NO READER OR WRITER TASK CURRENTLY ACTIVE			$GAM

IPW$$CD - PDISPLAY PNET Processor



NOTES	MODULE	LABEL	REF
1 Message 1RA3I ccccccccc VSE/POWER NETWORKING NOT SUPPORTED is issued.			$GAM
2 The Field CPWDID contains P,which indicates the PNET Command was issued.			
2D The network definition table is scanned to locate the entry for the specified node. If not found message 1RA41 ccccccccc nodename INVALID is issued.			$GAM
If none of above has been specified, message 1R521 ccccccccc OPERAND ## MISSING OR INVALID is issued.			$GAM
3 The 3rd operand specifies the printer device address on which the print status display should be printed out. The 'Assign' subroutine is called to check if the device is defined in the PUB, the device is operational, not used and a valid printer device. If so, a free LUB is assigned to the specified printer device, unless the initiator/terminator task has originated the command, in which case SYSLST is used as LUB.			

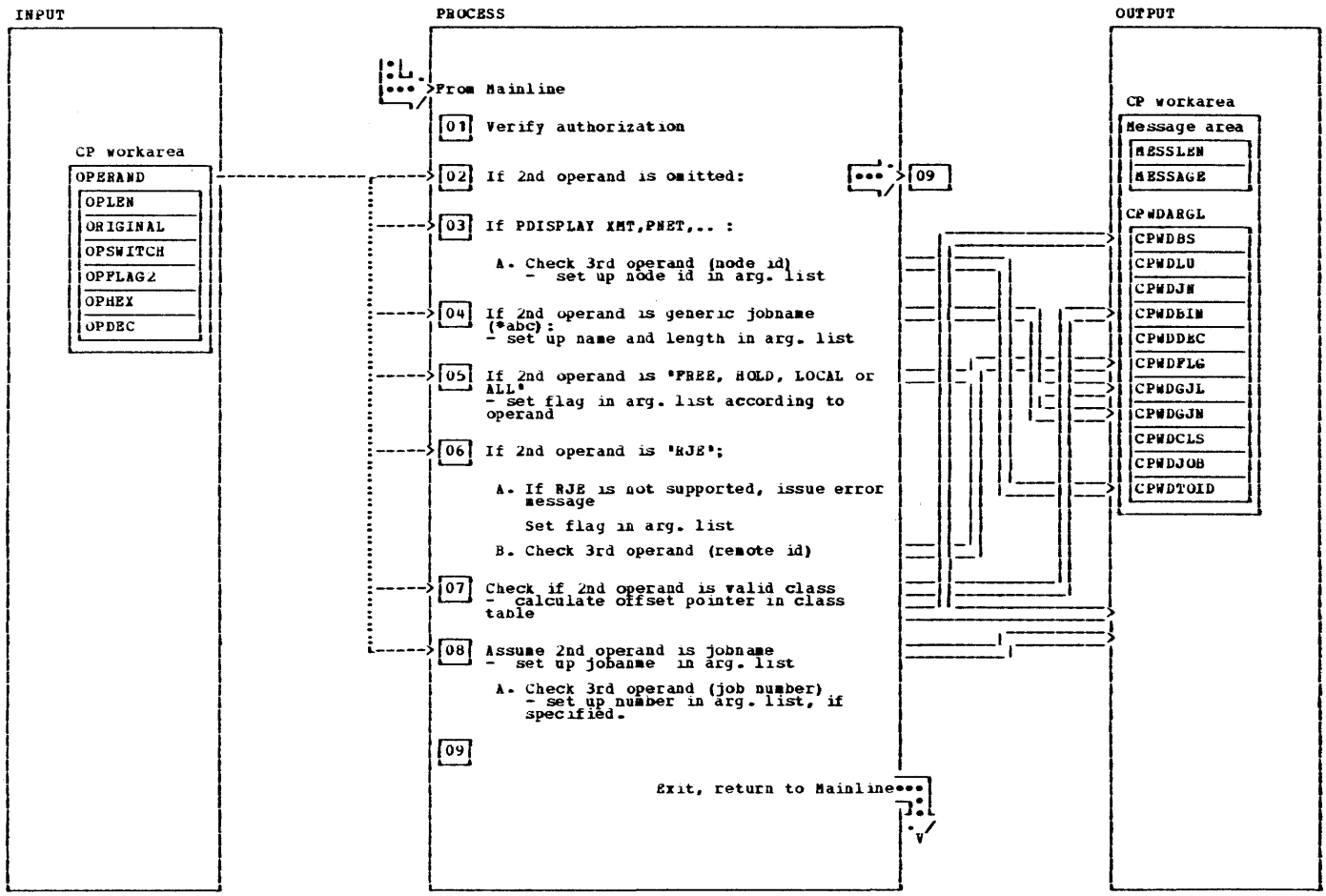
NOTES	MODULE	LABEL	REF
3 When an error is detected by the 'Assign' routine, an message is already put out by it.			
4 The dummy TCB area of the CP work area is used to set up the TCB for the new task. Following values are initialized: - task id ('P PS') - device address - Entry point address - remote identification - display argument list.			


Process Queue display
INPUT
PROCESS
OUTPUT
CP workarea
OPERAND
OPLEN
ORIGINAL
OPSWITCH
OPFLAG2
OPHEX
OPDEC
From Mainline
01 Clear argument list, used by IPW$$PS (Print Status display) - assume PDISPLAY ALL
02 If 1st operand is generic jobname (*abc):
Verify authorization
03 If 1st operand 'HOLD, FREE or LOCAL':
Verify authorization
Set flag in arg. list according to operand.
04 If 1st operand is 'RJE':
Verify authorization
A. Check 2nd operand, if specified (remote-id)
05 If 1st operand is 'ALL':
Verify authorization
A. Check 2nd operand (cuu), if specified
Assign printer device
06 If 1st operand Queue id:
A. Process queue display CD-1.5.1
07 Issue error message
08 Check keyword operands and set up argument list for queue scan routine:
- current disposition (CDISP)
- current priority (CPRI)
- current destination (CDEST)
- current system id (CSYSID)
If operand is wrong:..
09 Build TCB for Print Status task
10 Attach new task
A. Clear ECB address
11
Exit, return to Mainline
CP workarea
Message area
MESSLEN
MESSAGE
CPWDARGL
CPWDBS
CPWDLU
CPWDJN
CPWDBIN
CPWDDEC
CPWDFLG
CPWDGJL
CPWDGJN
CPWDCLS
CPWDJOB
CPWDTOID
CPWDCDP
CPWDCPY
CPWDCSY
Assign arg.list
CPWASDEV
CPWASCUU
CPWASLUN
DYSD
DYTI
DYCU
DYTT
DYPC
DYW1
TCB
CPEA

NOTES	MODULE	LABEL	REF
2 The 'Command authorization' subroutine is called to check if the issuer of the command is authorized. If not, the routine has already issued an appropriate error message and the command is ignored .			$VCA
3 The 'Command authorization' subroutine is called to check if the issuer of the command is authorized. If not, the routine has already issued an appropriate error message and the command is ignored .			$VCA
4 The 'Command authorization' subroutine is called to check if the issuer of the command is authorized. If not, the routine has already issued an appropriate error message and the command is ignored .			$VCA
4A The 2nd operand specifies the remote id, any number between 0 and 200. the number must be a valid VSE/POWER remote id and therefore defined in the PRMT generated remote table. Message 1R52I cccccccc OPERAND ## MISSING OR INVALID is issued, when the specification is wrong.			$GAM
5 The 'Command authorization' subroutine is called to check if the issuer of the command is authorized. If not, the routine has already issued an appropriate error message and the command is ignored .			$VCA
5A The 2nd operand specifies the printer device address on which the print status display should be printed out. The 'Assign' subroutine is called to check if the device is defined in the PUB, the device is operational, not used and a valid printer device. If so, a free LUB is assigned to the specified printer device, unless the initiator/terminator task has originated the command, in which case SYSLST is used as LUB.			

NOTES	MODULE	LABEL	REF
5A When an error is detected by the 'Assign' routine, an message is already put out by it.			
6 The 'Verify Queue id' subroutine is called to check if the operand is valid. Valid specification is either 'RDR, LST, PUN or XMT'.			
7 The first operand is wrong, following message will be issued: 1R52I cccccccc OPERAND ## MISSING OR INVALID			$GAM
8 The 'VERKEYO' subroutine is called to check if the keyword and the keyword value is valid. If not, the subroutine has already issued an appropriate error message.			
9 The dummy TCB area of the CP work area is used to set up the TCB for the new task. Following values are initialized: - task id ('P PS') - device address - Entry point address - remote identification - display argument list.			

Process Queue display

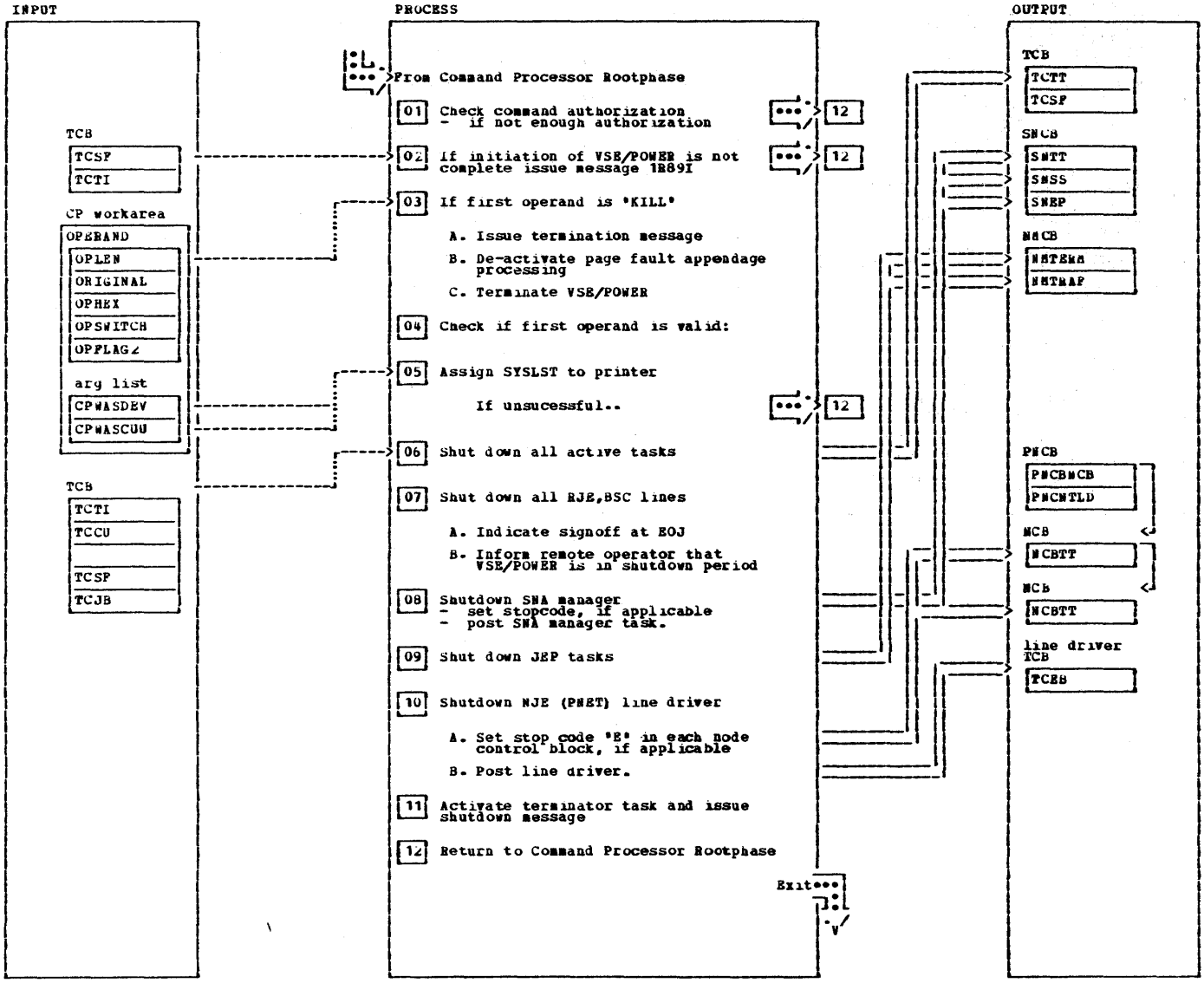


	NOTES	MODULE	LABEL	REF
1	The 'Command authorization' subroutine is called to check if the issuer of the command is authorized. If not, the routine has already issued an appropriate error message and the command is ignored .			$VCA
3A	The 3rd operand specifies the destination nade name. Only jobs/ouput destined for that node should be displayed. The node name must be alphameric, first character alphabetic, up to eight bytes long. If the specification is invalid, message IR52I cccccccc OPERAND ## MISSING OR INVALID is issued.			$GAM

	NOTES	MODULE	LABEL	REF
3A	No check is made at this point, to ensure that the specified node node is defined in the noework definition table.			
6B	The 3rd operand specifes the remote id of the queue sets (either to or from). Only queue sets owned by the remote id are displayed. If the specifiaction is not numeric, in the range between 0 - 200, message IR52I cccccccc OPERAND ## MISSING OR INVALID is issued.			$GAM

This page was left blank intentionally.

IPW$$CE - PEND Processor

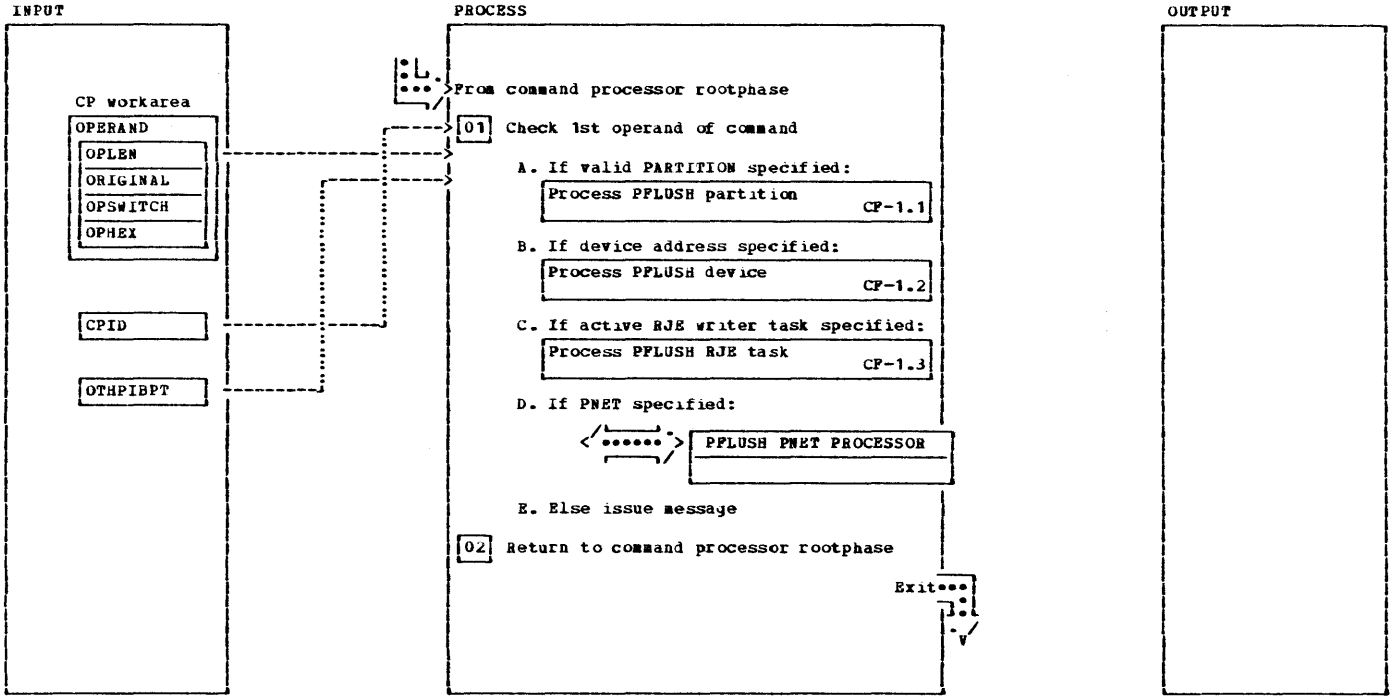


IPW$$CE - PEND Processor

NOTES	MODULE	LABEL	REF
The PEND command has the following formats either PEND <urr> or PEND KILL where is: urr : Specifies a printer device address KILL : Specifies an immediate termination		PEND000	
1 The authority associated to the issuer of the command is examined. This is done by invoking the appropriate subroutine located in the Rootphase of the command processor.			$VCA
2 Message 1R89I VSE/POWER INITIATION NOT COMPLETE is issued.			$GAM
3A Following message will be issued: 1R99I VSE/POWER HAS BEEN TERMINATED			$GAM
3B The page fault appendage routine of VSE/POWER is de-activated.	SETPFA		
3C Control is passed to the VSE/POWER abnormal termination exit routine directly.			$CNC
4 The first operand is the address of a printer device. If the specification is invalid message is issued: 1R52I command code OPERAND ## MISSING OR INVALID			$GAM

NOTES	MODULE	LABEL	REF
5 In case of unsuccessful assignment subroutine ASSIGN has already issued error messages, so a branch is taken to processor exit			
6 Each task which is in the task selection list is examined and depending on the task type following termination codes are set in the appropriate TCB: - for all RJE task, except of SNA and JEP, term code 'E' - depending on the state of an execution reader task either stop code 'S' or 'E' is set. 'S' is set if the task is waiting for work (Q-state, S-state), but never for an execution reader of a writer only partition. - for physical task (LST,PUN or RDR) stop code 'E' is set unless the task is waiting for work (Q-state), inactive (I-state) or in S-state, in which case stop code 'S' is set.			
7B All signed-on remote operators are informed that VSE/POWER is in shutdown period by means of message 1R99I.			$GAM
11 Finally the initiator task is modified into terminator task and is then reactivated			
11 Following message will be issued: 1R99I VSE/POWER IS IN THE SHUTDOWN PERIOD			

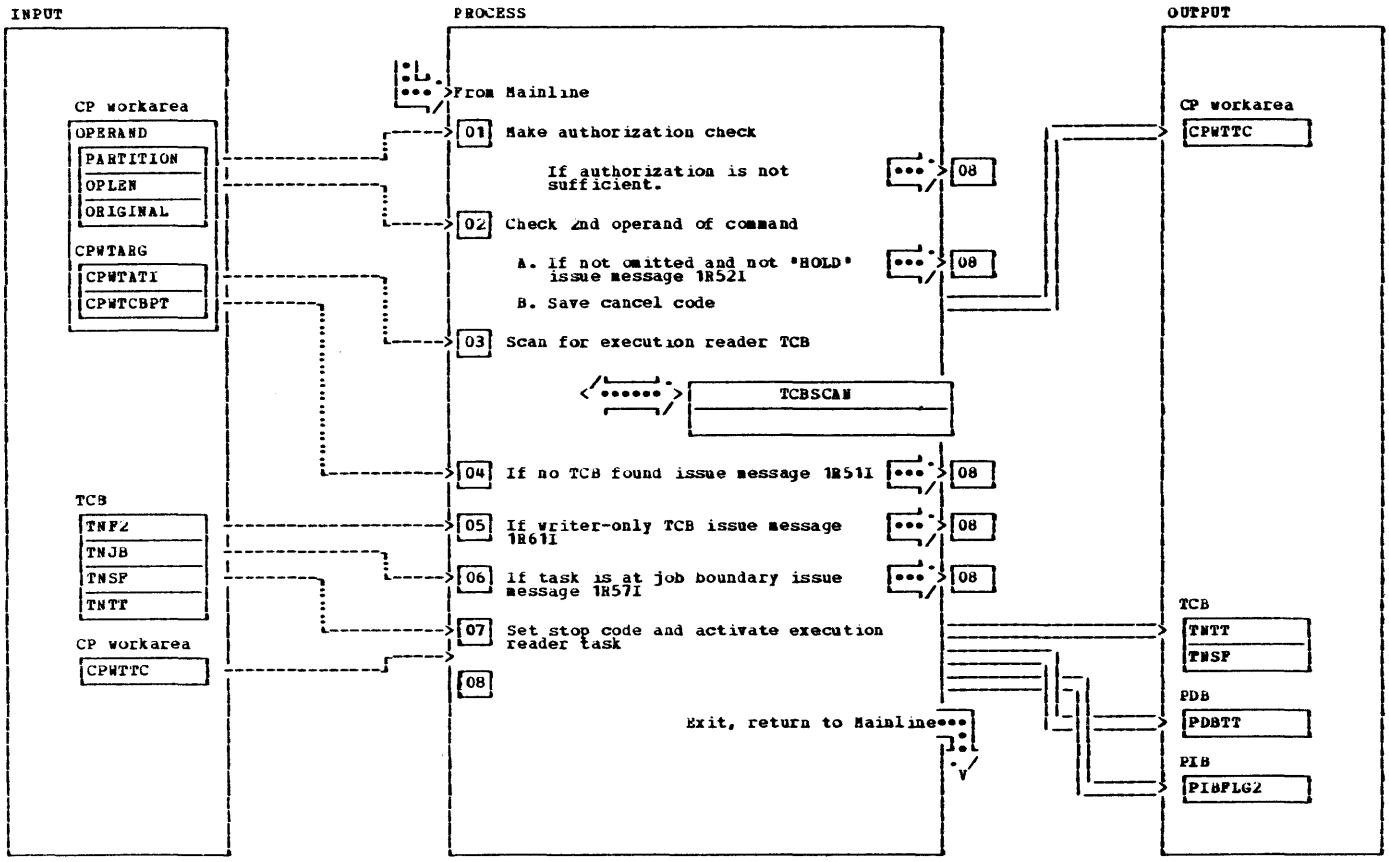
IPW$$CF - PFLUSH Processor



NOTES	MODULE	LABEL	REF
1 The PFLUSH command has the following format either PFLUSH uraddr<,HOLD> or PFLUSH PARTITION<,HOLD> or PFLUSH task<,HOLD> if entered from remote where ,OUT PFLUSH PNET,nodeid,RVn,JOB ,OUT PFLUSH PNET,nodeid,TRn,JOB<,HOLD> uraddr : Specifies a device address in the form cuu or x'cuu'. PARTITION : Specifies a PARTITION from BG to Fn (n = PARTITION number). task : Specifies an active RJE writer task LST,LST1,LST2 LST3 or PUN note : This operand is only valid for a FLUSH command entered by a remote operator. PNET : Specifies that Partition Network is active . nodeid : Specifies the node id for the task to be flushed. RVn : Specifies the Receiver line # (n = 1 thru 7). TRn : Specifies the Transmitter line # (n = 1 thru 7).			

NOTES	MODULE	LABEL	REF
JOB : Specifies that a Transmitter or Receiver task could be a job. Out : Specifies that a Transmitter or Receiver task could be output. HOLD : Specfies that the que entry for a Transmitter task only is not to be deleted but place in a hold state. 1 after return from subroutine VPARTID the field OTHPIBPT will be zero if an invalid partition id has been specified 1B a valid device address is one in which oplen = 3 and the first operand is hex. Else if oplen ¬=3 then issue message: 1R52I command code OPERAND ## MISSING OR INVALID 1C for writer task only: cpid¬ = 0 (remote operator command) and oplen = 3. else issue message: 1R52I command code OPERAND ## MISSING OR INVALID validate task id by calling subroutine vtaskid - if not valid issue message: 1R52I command code OPERAND ## MISSING OR INVALID 1E			$GAM

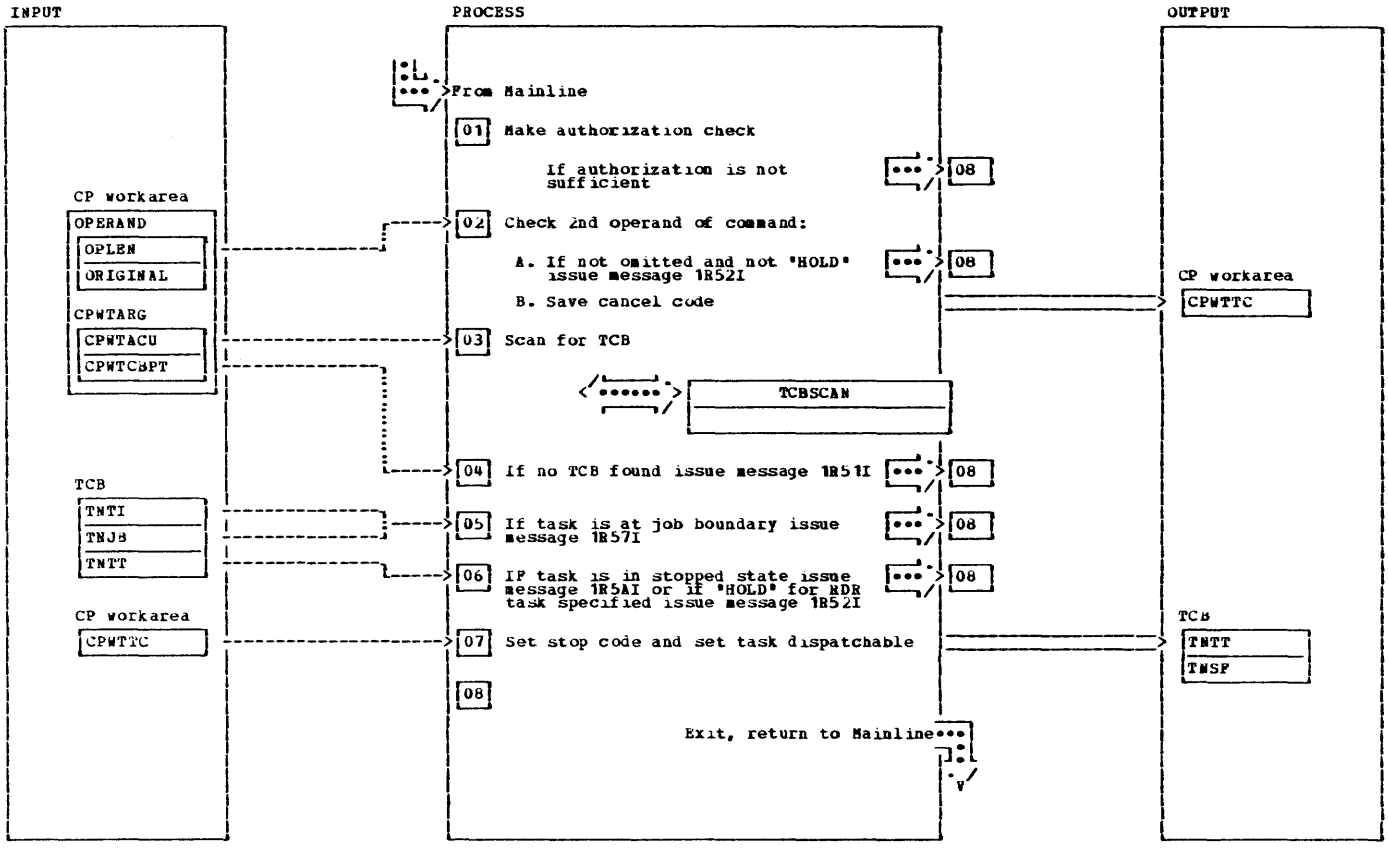
IPW$$CP - PPLUSH,PARTITION Processor



NOTES	MODULE	LABEL	REF
1 Issue IPW$VCA macro to check authorization			$VCA
2 If 2nd operand is omitted assume termination code 'F' or if 'HOLD' specified assume 'H'			
2 Else following message will be issued: 1R52I command code OPERAND ## MISSING OR INVALID			
3 Set up following fields before invoking subroutine TCBSCAN: CPWTARG = ' ', CPWTATI(2) = 'E ', CPWTATI+2(2) = PARTITION-id			
4 If no TCB found as requested CPWTCBPT contains zeroes instead of a valid TCB address			
4 Following message will be issued: 1R51I command code OPERAND ## DESIGNATES NON-EXISTING TASK			

NOTES	MODULE	LABEL	REF
5 If a matching TCB found, the field TCDT contains the device type code, e.g. x'00' or x'B2' for writer-only TCB			
5 ELSE following message will be issued: 1R61I command code INVALID FOR WRITER ONLY PARTITION			
6 IF task is at job boundary TCJB is not x'FF'			
6 Else following message will be issued: 1R57I command code IGNORED, TASK IS AT JOB BOUNDARY			
7 Set stop code in TCTT of TCB. Save original termination code in PDB. Set new stop code into TCB, get PIB of appropriate PARTITION and if partition in stopped state activate ECB			

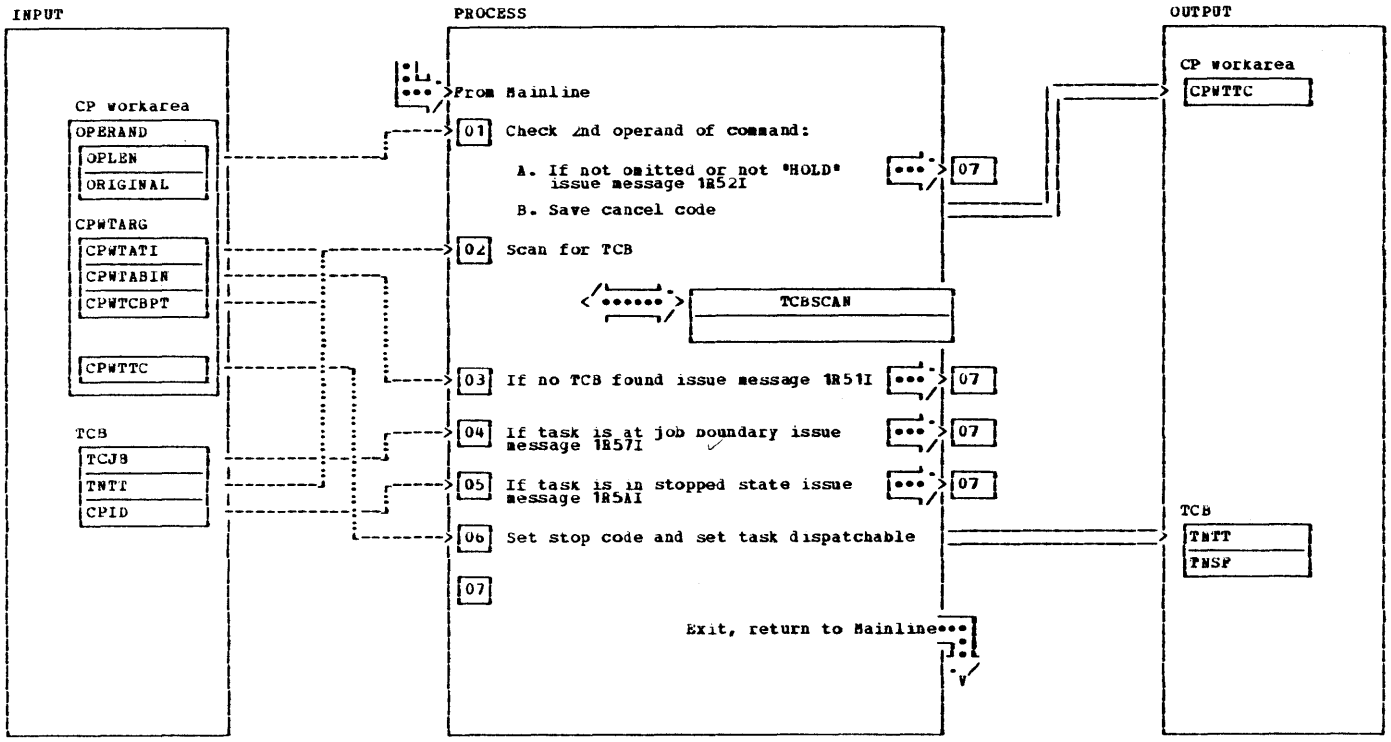
IPW$$CF - PFLUSH,device Processor



NOTES	MODULE	LABEL	REF
1 Issue IPW$VCA macro to check authorization			$VCA
2 If 2nd operand is omitted assume termination code 'P' and if 'HOLD' specified assume 'H'			
2 ELSE following message will be issued: 1R52I command code OPERAND ## MISSING OR INVALID			
3 Set up following fields before invoking subroutine TCBSCAN: CPWTARG = ' ', CPWTACU(1) = ' ', CPWTACU+1(3) = device address CPWTAFLG = CPWTACNT, to force a complete scan thru TCB chain			

NOTES	MODULE	LABEL	REF
4 If no TCB found as requested CPWTCBPT contains zeroes instead of a valid TCB address			
4 Following message will be issued: 1R51I command code OPERAND ## DESIGNATES NON-EXISTING TASK			
5 IF task is at job boundary TCJB is not x'FF'			
5 Following message will be issued: 1R57I command code IGNORED, TASK IS AT JOB BOUNDARY			
6 Following messages will be issued: 1R52I command code OPERAND ## MISSING OR INVALID or 1R5AI command code FLUSH IGNORED, TASK IS IN STOP STATE			
7 Set stop code and set task dispatchable			

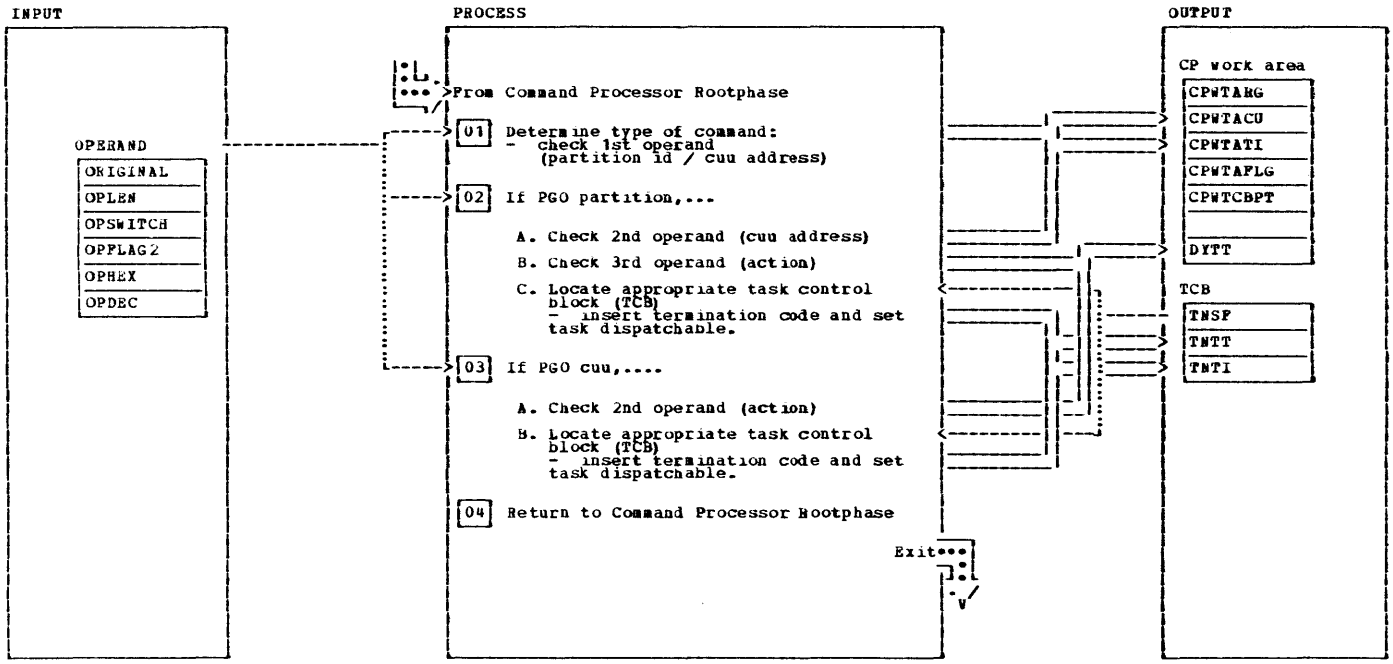
IPW$$CP - PPLUSH,RJE writer task, Processor



NOTES	MODULE	LABEL	REF
1 If 2nd operand is omitted assume termination code 'P' and if 'HOLD' specified assume 'h'			
1 Else following message will be issued: 1R52I command code OPERAND ## MISSING OR INVALID			
2 Set up following fields before invoking subroutine TCBSCAN: CPWTARG = ' ', CPWTATI(1) = '1', CPWTATI+1(3) = task specification, CPWTABIN = CPID			
3 If no TCB found as requested CPWTCBPT contains zeroes instead of a valid TCB address			

NOTES	MODULE	LABEL	REF
3 Following message will be issued: 1R51I command code OPERAND ## DESIGNATES NON-EXISTING TASK			
4 IF task is at job boundary TCJB is not x'FF'			
4 Following message will be issued: 1R57I command code IGNORED, TASK IS AT JOB BOUNDARY			
5 Following messages will be issued: 1R52I command code OPERAND ## MISSING OR INVALID or 1R5AI command code FLUSH IGNORED, TASK IS IN STOP STATE			
6 Set stop code and set task dispatchable			

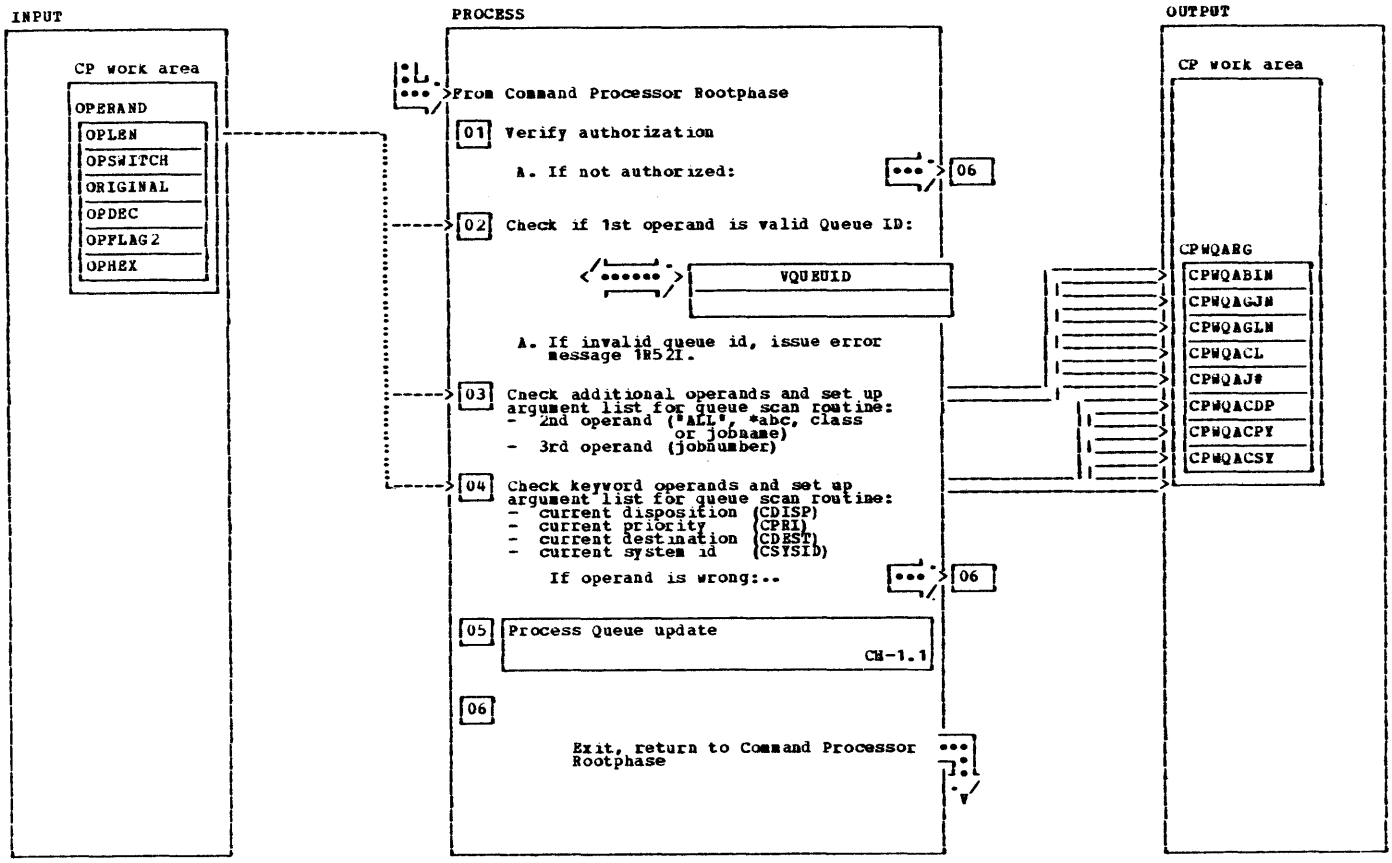
IPW$$CG - PGO Processor



	NOTES	MODULE	LABEL	REF
	The PGO command has two formats: PGO partition,cuu<,action> or PGO cuu<,action> where: partition: One of BG,FA,FB,F1 thru F9 cuu : A valid unit record or tape device address in the form cuu or x'cuu' action : Either CANCEL,IGNORE or UNLOAD			
1	If the first operand is not a device address in the cuu format, the 'VPARTID' subroutine is called to check if the 1st operand is a valid partition-id supported by DOS/VSE.		PGO100	
2A	If the operand is not a valid device address, message 1R51I command code OPERAND ## NO DEVICE ADDRESS is issued.		PGOP200	$GAM
2B	The 3rd operand specifies the action to be taken. Valid specifications are 'UNLOAD, CANCEL or IGNORE'. The operand is optional. If none of above has been specified, message 1R52I command code OPERAND ## MISSING OR INVALID is issued.		PGOP300	$GAM
2C	The 'TCBSCAN' subroutine is called in order to locate the TCB to be re-activated.		PGOP800	
2C	After return from subroutine TCBSCAN the field CPWTCBPT contains the address of the TCB found or if no TCB could be located it contains zeroes. In the later case the following message is issued: 1R51I NON-EXISTING TASK DESIGNATED Otherwise the TCB found is checked on wait state. If this is not true the following message is issued: 1R77I command code TASK NOT WAITING FOR OPERATOR.			

	NOTES	MODULE	LABEL	REF
2C	The termination type is set according to the action operand and the task is posted dispatchable.			
3A	The 2nd operand specifies the action to be taken. Valid specifications are 'UNLOAD, CANCEL or IGNORE'. The operand is optional. If none of above has been specified, message 1R52I command code OPERAND ## MISSING OR INVALID is issued.		PGOC200	$GAM
3B	The 'TCBSCAN' subroutine is called in order to locate the TCB to be re-activated.		PGOC800	
3B	After return from subroutine TCBSCAN the field CPWTCBPT contains the address of the TCB found or if no TCB could be located it contains zeroes. In the later case the following message is issued: 1R51I NON-EXISTING TASK DESIGNATED Otherwise the TCB found is checked on wait state. If this is not true the following message is issued: 1R77I command code TASK NOT WAITING FOR OPERATOR.			
3B	The termination type is set according to the action operand and the task is posted dispatchable. Following termination codes are set: no action operand - ' ' 'UNLOAD' - 'N' 'IGNORE' - 'I' 'CANCEL' - 'S'			

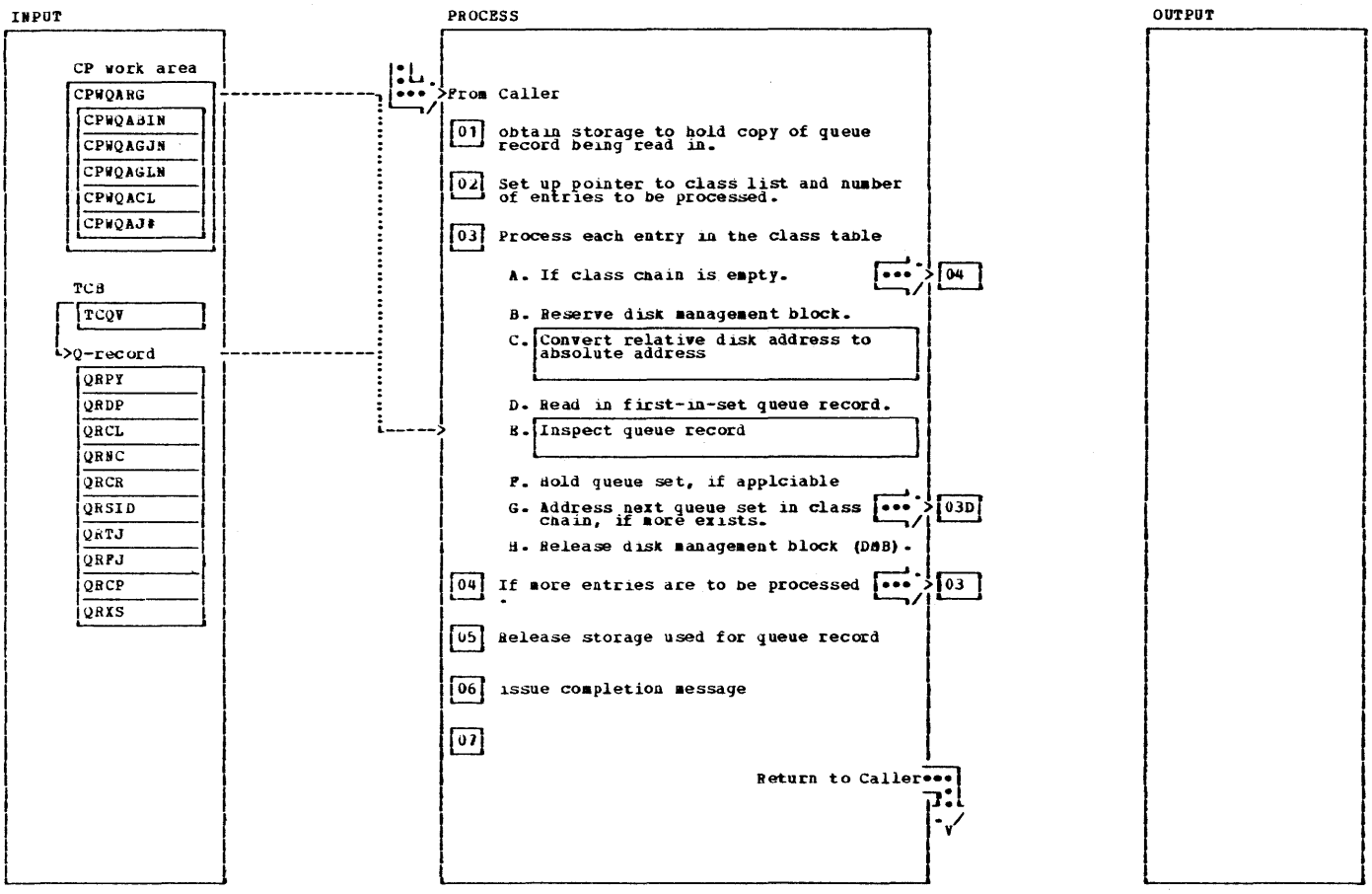
IPW$$CH - PHOLD Processor



NOTES	MODULE	LABEL	REF
The PHOLD command has the following possible formats: PHOLD queue,jobname<,jobnumber> <,keyword=value> PHOLD queue,ALL <,keyword=value> PHOLD queue,class <,keyword=value> PHOLD queue,*abc <,keyword=value> where: queue : One of LST,RDR,PUN or XMT jobname : A job name known to VSE/POWER jobnumber: A job number assigned to job name ALL : Specifies that all jobs in queue are to be held class : Specifies that all jobs in queue for this class are to be held keyword : One of CPRI, CDEST CDISP or CSYSID value : specifies current value of the attribute *abc : Specifies that all jobs in the queue, whose names have the specified characters in common, are to be held (generic job name)		PHLD000	

NOTES	MODULE	LABEL	REF
1 The 'Verify command authorization' subroutine is called to check if the issuer of the command is authorized. If not, the command is rejected. An appropriate error message has been already issued by the subroutine.			$VCA
2 The 'Verify Queue id' subroutine is called to examine if the operand specifies a valid queue id. Valid queue ids are - RDR, LST, PUN or XMT.			
2A Message 1R52I cccccccc OPERAND 1 NO VALID QUEUE is issued.		PHLD100	$GAM
3 If invalid specifications have been entered message 1R52I cccccccc OPERAND ## MISSING OR INVALID will be issued.			$GAM
4 The 'VERKEYO' subroutine is called to check if the keyword and the keyword value is valid. If not, the subroutine has already issued an appropriate error message.			
5 The RDR, LST, PUN and/or XMT queue is scanned for queue set(s) specified by operand 2 and 3. Each queue set which is eligible (meeting the criteria) with a disposition of 'D' or 'K' is put in 'hold' state. The disposition is set to 'H' if the original disposition is 'D' or to 'L' if the original disposition is 'K'.			

IPW$$CH - Process Queue update

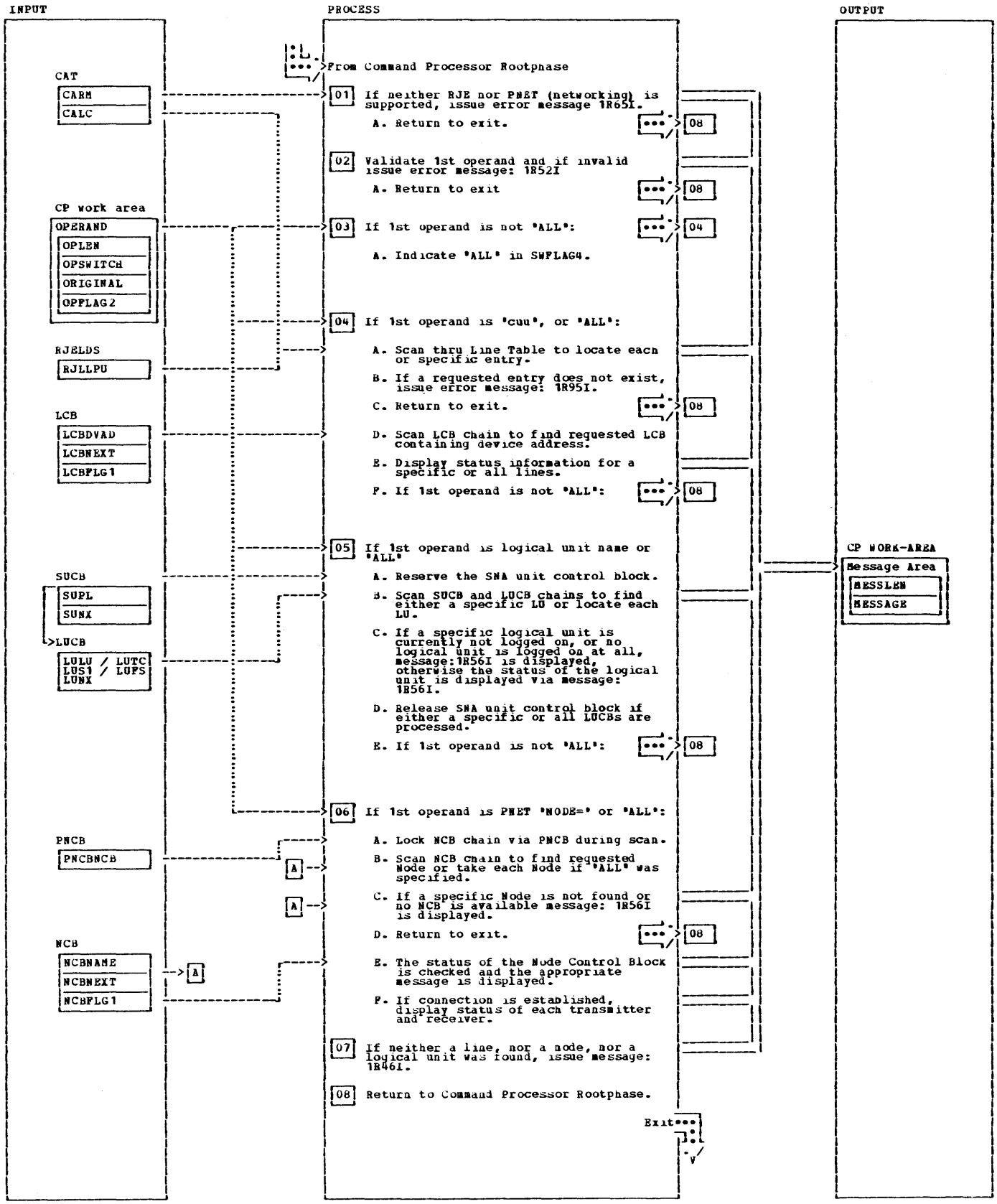


	NOTES	MODULE	LABEL	REF
	The reader , list , punch and/or xmit queue is scanned and each queue set which is eligible is extracted and updated.			
1	For the queue record area workspace will be provided.		PHLD500	$RSW
3B	The disk management block is reserved for the duration of the queue scanning.			$RSR
3C	Suproutine RELTOABS converts the relative disk address			
3D	The first in set queue record is read in the queue record area just acquired.			$RDQ
3E	The queue record just read in is examined if it is eligible to be updated. - jobname and number - class - RJE user id - local or remote destination			

	NOTES	MODULE	LABEL	REF
3F	When the disposition of the queue set is either 'D' or 'K' it is changed to 'H' or 'L' respectively. Then the first-in-set queue record is written back.			$WTQ
3G	Queue sets belonging to a class are chained via the 'QCQN' field. This field contains the absolute disk address of the next queue set in chain. For the last queue set, this field contains binary zeros.			
3H	The disk management block is released.			$RLR
5				$RLW
6	If any queue set is held message 1R88I OK for the central operator only or if not 1R88I NOTHING TO HOLD will be issued.			$GAM

This page was left blank intentionally.

IPW$$CI - PINQUIRE Processor

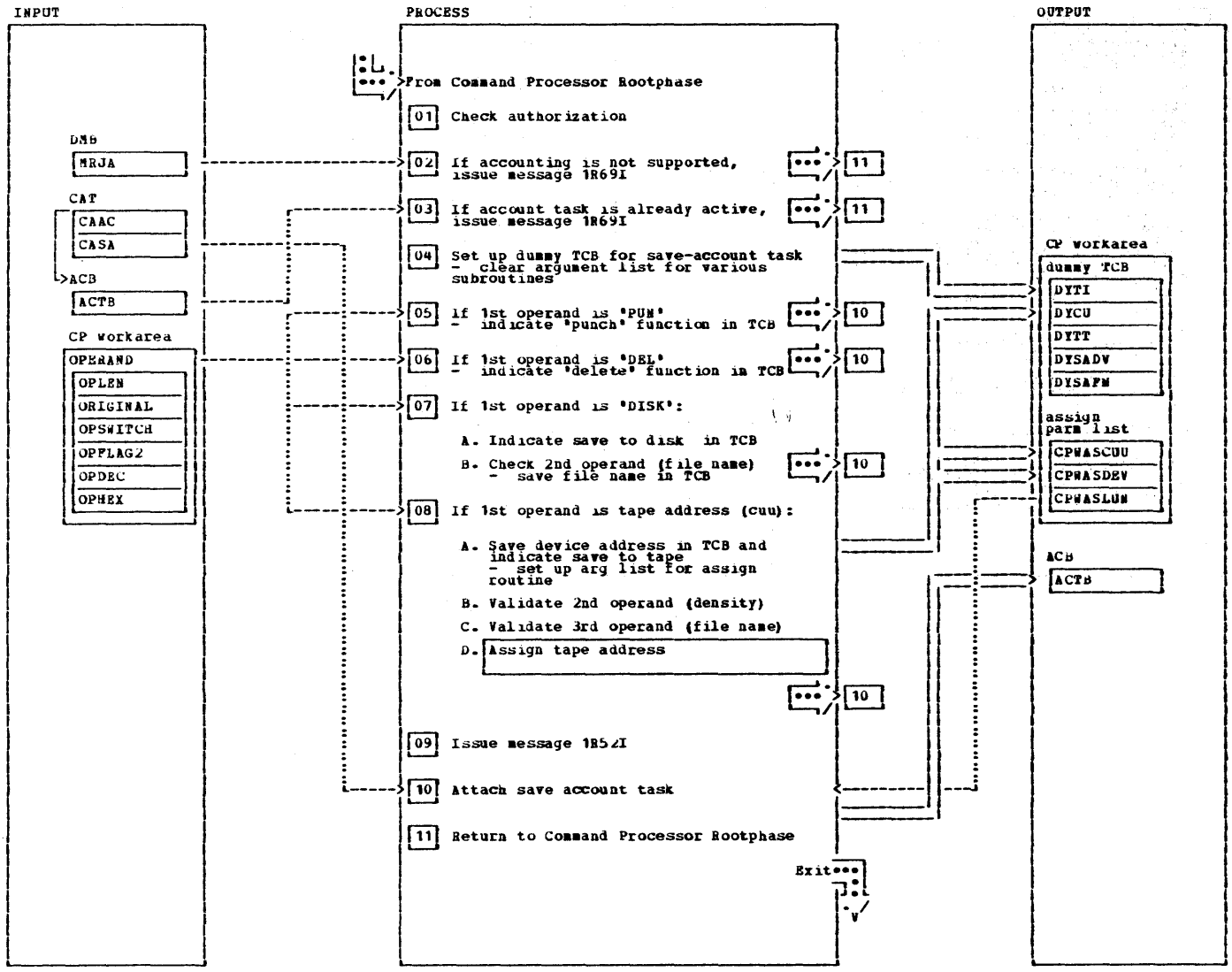


IPW$$CI - PINQUIRE Processor

	NOTES	MODULE	LABEL	REF
	The function of this command is to display status information for RJE BSC lines, RJE SNA logical units, or PNET Nodes. The status of a RJE BSC line may be: 1.) Not supported, because no line table entry exists. 2.) Not initiated, because no line control block exists. 3.) Inactive, when a line Control Block exists, but nobody is signed on yet. 4.) Processing remote-id nnnnnnnn, somebody is signed on. The status of a RJE SNA logical unit may be: 1.) Not logged on, because no SUCB exists for this particular logical unit. 2.) Logging on, the logical unit is in the middle of the logon process. 3.) processing remote id. The status of PNET Node may be: 1.) Node not logged on, because no NCB exists for this particular Node. 2.) Session pending, because a Node Control Block exists but the other end has not logged on. 3.) Node processing. The PINQUIRE command has one of the following possible formats: PINQUIRE lineaddr PINQUIRE luname PINQUIRE NODE=nodename PINQUIRE ALL where: lineaddr : BSC line address in the form cuu or x'cuu' luname : Name of a SNA logical unit nodename : Name of PNET Node ALL : All lines and/or lunames and/or sessions or remote id's are to be checked			
1	Following message is issued: 1R65I cccccccc RJE NOT SUPPORTED and control is given back.		PINQMSG	$GAM
2	The following message is displayed: 1R52I cccccccc OPERAND ## MISSING OR INVALID.		PINQMSG	$GAM
3	If 'ALL' is specified, all BSC Lines, all SNA Logical Units, and all PNET Nodes are checked and their status is displayed.			
4A	The Line Table is scanned to find either a specific entry or to locate each entry if 'ALL' was specified.			
4B	If no specific entry was found, message: 1R95I cccccccc LINE cuu NOT SUPPORTED. is displayed and control is given back.		PINQMSG	$GAM
4D	The LCB chain is scanned to find a corresponding LCB to the line address.			
4E	Following messages will be issued: If no LCB exists, message: 1R56I xxx NOT INITIATED is displayed. Otherwise, the following messages are displayed: 1R56I cuu PROCESSING rrr or 1R56I cuu INACTIVE		PINQMSG	$GAM
5A	The SUCB chain is locked for the duration time of the scan.			$RSR
5B	Each SUCB is taken and the related LUCB's are scanned to find either a specific LU or to take each LU if 'ALL' was specified.			
5B	The following messages are displayed: 1R56I xxx PROCESSING yyy or 1R56I xxx LOGGED ON yyy or 1R56I xxx LOGGING ON yyy or if not logged on 1R56I NO LOGICAL UNIT LOGGED ON 1R56I xxx NOT LOGGED ON		PINQMSG	$GAM
5D	Release SNA Unit Control Block.			$RLR
5	If a logical unit is specified, status information of a specific SNA logical unit is displayed.			
6A	The PNCB is locked for the duration time of the scan.			$RSR
6B	The NCB chain is scanned to locate the appropriate node control block.			
6C	The following message is displayed: 1R56I NODE nnnnnnnn NOT LOGGED ON 1R56I NO LOGICAL UNIT LOGGED ON.		PINQMSG	$GAM
6E	The status of a Node can be: 1R56I NODE nnnnnnnn SESSION PENDING 1R56I NODE nnnnnnnn PROCESSING		PINQMSG	$GAM
6F	The status of an individual receiver/transmitter can be either: - active, if the task is just receiving or sending - inactive, if the task is at job boundary - drained.		PINQMSG	$GAM
7	If nothing to display, message 1R46I NOTHING TO DISPLAY is issued.		PINQMSG	$GAM

This page was left blank intentionally.

IPW$$CJ - PACCOUNT Processor

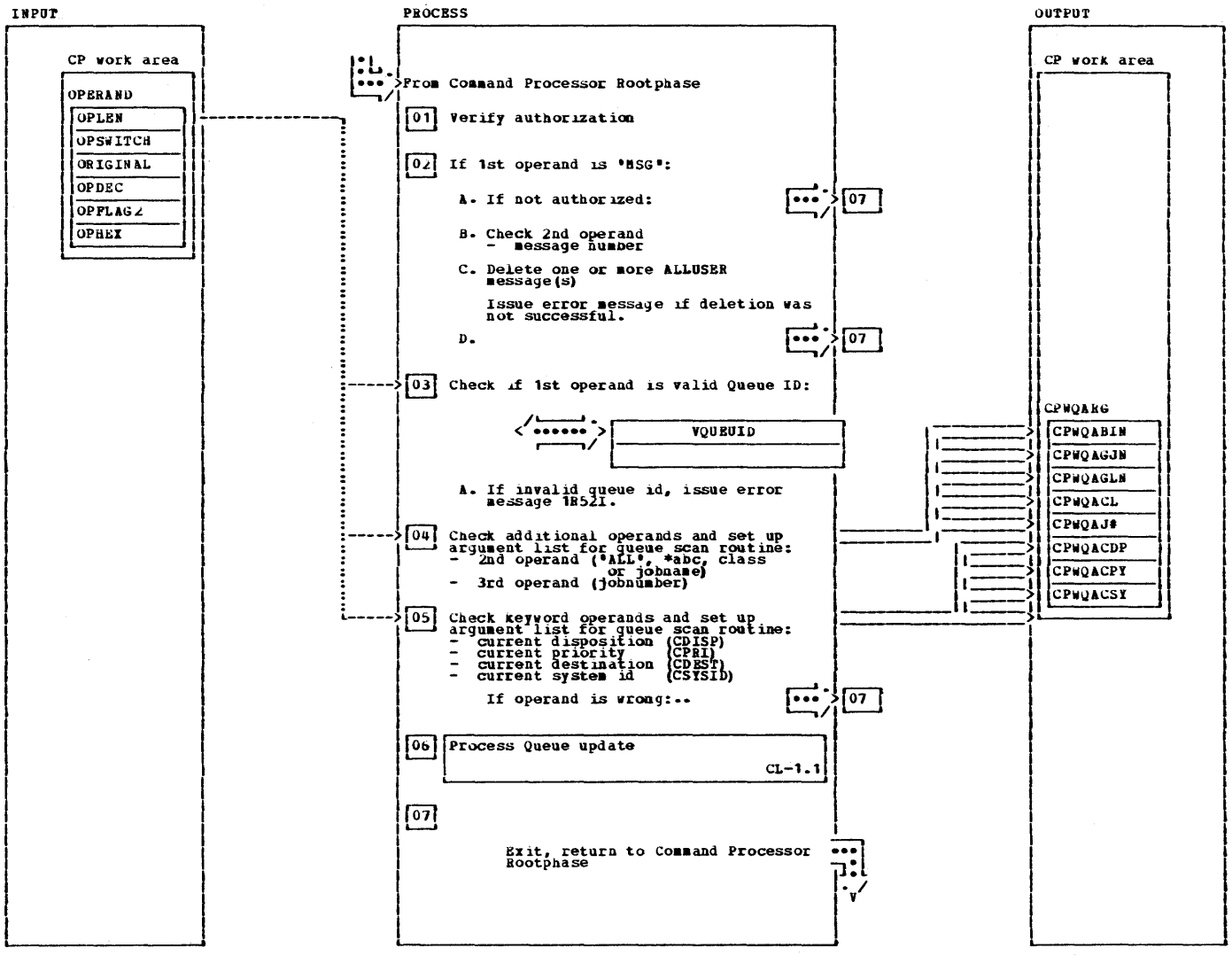


IPW$$CJ - PACCOUNT Processor

NOTES	MODULE	LABEL	REF
The PACCOUNT command has the following format either PACCOUNT <tapeaddr>,<density><,tfilename> or PACCOUNT DISK,dfilename or PACCOUNT DEL or PACCOUNT PUN where is: tapeaddr : Specifies a tape device address in the form cuu or x'cuu' density : Density of output tape in the form ss or x'ss' tfilename : Filename from //TLBL dfilename : Filename from //DLBL		PACC000	
1 The 'command authorization' subroutine is called to check if the issuer of the PACCOUNT command is authorized. If the issuer has not enough authorization, the command is rejected. The subroutine has already put out an appropriate error message, therefore a branch is made to the exit.			$VCA
2 Following message will be issued: 1R691I cccccccc NO ACCOUNTING SUPPORT			$GAM
3 Following message will be issued : 1R691I cccccccc COMMAND REJECTED, SAVE ACCOUNT ALREADY ACTIVE			$GAM
4 Task identifier is ' ACT'.			
7B The 2nd operand specifies the file name of the disk extent to be used. The operand is mandatory. If omitted, message 1R52I cccccccc OPERAND ## MISSING OR INVALID is issued.			$GAM
7B If the operand is not alphameric, first character alphabetic, message 1R55I cccccccc INVALID FILENAME is issued.			$GAM
8A If the tape address is not defined as valid tape device message 1R74I cccccccc INVALID DEVICE SPECIFICATION is issued.			$GAM
8B The 2nd operand specifies the density to be used. If the operand is omitted, the default density according the device type is used. Otherwise the PUB is scanned to locate the appropriate device entry. If no entry is found, message 1R58I cccccccc DEVICE cuu IS NOT KNOWN is issued.			$GAM
8B If the specified density is invalid according to the 7/9 track tables, message 1R53I cccccccc INVALID DENSITY is issued.			$GAM
8C If the file name specification is invalid, message 1R55I cccccccc INVALID FILENAME is issued.			$GAM
8D In case of unsuccessful assignment, subroutine ' ASSIGN' has has already issued error messages, so a branch is taken to processor exit			
9 the 1st operand of the PACCOUNT command is invalid, message 1R52I cccccccc OPERAND ## MISSING OR INVALID is issued.			$GAM

This page was left blank intentionally.

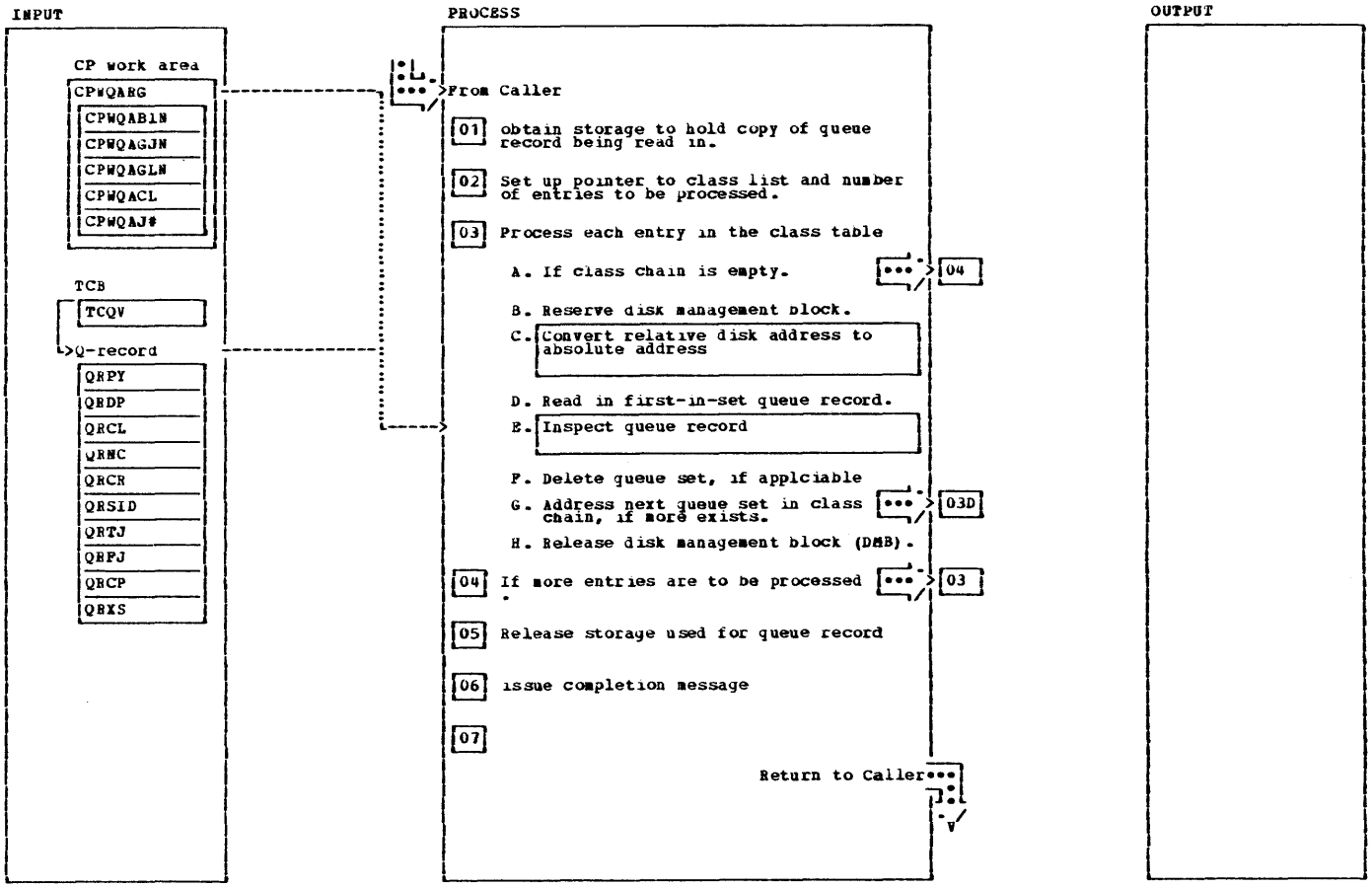
IPW$$CL - PDELETE Processor



NOTES	MODULE	LABEL	REF
The PDELETE command has the following possible formats: PDELETE queue<,jobname<,jobnumber>> <,keyword=value> PDELETE queue<,ALL> <,keyword=value> PDELETE queue<,class> <,keyword=value> PDELETE queue<,*abc> <,keyword=value> PDELETE MSG<,n>			
where: queue : One of LST,RDR,PUNor XMT jobname : A job name known to VSE/POWER jobnumber: A job number assigned to job name ALL : Specifies that all jobs in queue are to be deleted class : Specifies that all jobs in queue for this class are to be deleted *abc : Specifies that all jobs in the queue whose names have the specified characters in common are to be deleted (generic jobname) keyword : One of CPRI, CDEST CDISP or CSYSID value : specifies current value of the attribute MSG : Specifies that messages are to be deleted n : Specifies that ALLUSER type messages numbered n are to be deleted			
2B The second operand is either numeric, specifying which ALLUSER message should be deleted or 'ALL' in which case all ALLUSER messages originated from the command issuer are to be deleted. If an invalid specification has been entered, message 1R52I ccccccccc OPERAND ## MISSING OR INVALID is issued.			$GAM

NOTES	MODULE	LABEL	REF
2C 2C message 1R84I DELETION NOT ALLOWED OR IMPOSSIBLE is issued			$xMS $GAM
1 The 'Verify command authorization' subroutine is called to check if the issuer of the command is authorized. If not, the command is rejected. An appropriate error message has been already issued by the subroutine.			$VCA
3 The 'Verify Queue id' subroutine is called to examine if the operand specifies a valid queue id. Valid queue ids are - RDR, LST, PUN or XMT.			
3A Message 1R52I ccccccccc OPERAND 1 NO VALID QUEUE is issued.		PDLT100	$GAM
4 If invalid specifications have been entered message 1R52I ccccccccc OPERAND ## MISSING OR INVALID will be issued.			$GAM
5 The 'VERKEYO' subroutine is called to check if the keyword and the keyword value is valid. If not, the subroutine has already issued an appropriate error message.			
6 The RDR, LST, PUN and/or XMT queue is scanned for queue set(s) specified by operand 2 and 3. If the queue record is eligible to be deleted, it is removed from the class chain			

IPW$$CL - Process Queue update

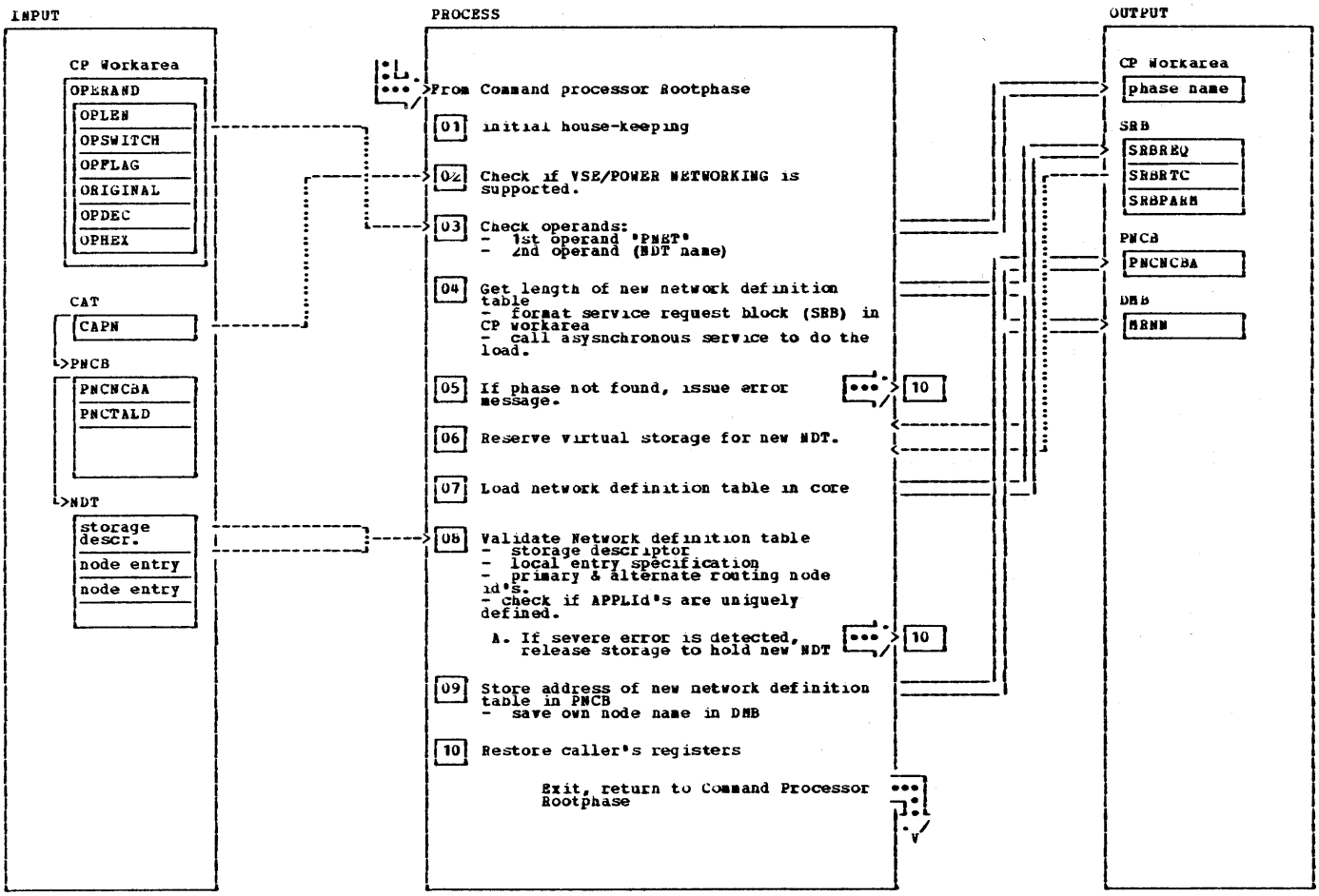


NOTES	MODULE	LABEL	REF
The reader , list , punch and/or xmit queue is scanned and each queue set which is eligible is extracted and updated.			
1 For the queue record area workspace will be provided.		PDLT500	$RSW
3B The disk management block is reserved for the duration of the queue scanning.			$RSR
3C Subroutine RELTOABS converts the relative disk address			
3D The first in set queue record is read in the queue record area just acquired.			$RDQ
3E The queue record just read in is examined if it is eligible to be updated. - jobname and number - class - RJE user id - local or remote destination			
3F If the queue set not currently being processed (execution switch on) the queue set is deleted from its class chain according to its disposition.		PDLT550	$DQS

NOTES	MODULE	LABEL	REF
3F All queue records, belonging to the queue set being deleted, are added to the free queue record chain.			$FQS
3G Queue sets belonging to a class are chained via the 'QCQN' field. This field contains the absolute disk address of the next queue set in chain. For the last queue set, this field contains binary zeros.			
3H The disk management block is released.			$MLR
5			
6 If any queue set is dletetd message 1R88I OK (for the central operator only) or if not 1R88I NOTHING TO DELETE is issued.		PDLT700	$RLW $GAM

This page was left blank intentionally.

IPW$$CLD - PLOAD NET=.... Processing



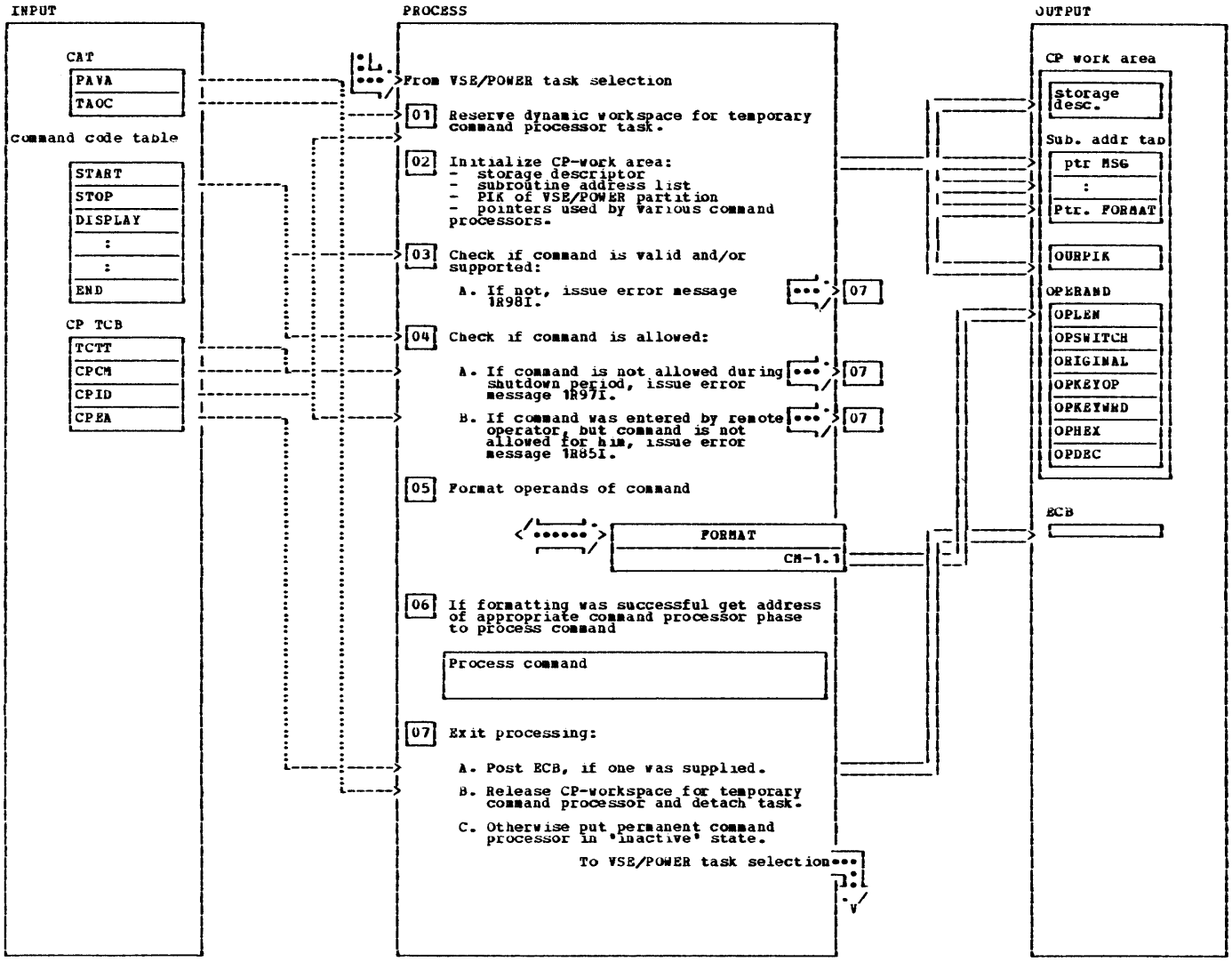
IPW$$CLD - PLOAD NET=.... Processing

NOTES	MODULE	LABEL	REF
1 The Command processor rootphase registers are stored in the save area reserved for this purpose. Addressability to various control blocks is set up.			
2 First of all a check is made if VSE/POWER networking is supported. If not, error message: 1RA3I VSE/POWER NETWORKING NOT SUPPORTED is issued.			
4 Asynchronous service is employed to do the load with the TXT=NO option in order to get the total length of the phase. Therefore a service request block is formatted and asynchronous service is called.			$IAS
6 Virtual storage is acquired to hold the new network definition table (NDT).			$RSV
6 If no storage can be obtained, message 107AI UNABLE TO LOAD NETWORK DEFINITION TABLE is issued.			$GAM
7 The service request block is updated to contain now the address where to load the network definition table in storage. Asynchronous service is called to the actual load.			$IAS
8 First of all the just loaded network definition table is examined if the table is valid: - storage descriptor and release level must match. - An own entry must be specified in the NDT. - The local node name of the new NDT must be identical to the current local node name. All prime and alternate routing node id's are examined if the appropriate node is defined in the network. If either the prime or alternate node id is not defined, message 1RA4I xxxxxxxx IS NOT A VALID NODEID is issued and the appropriate routing is nullified.			

NOTES	MODULE	LABEL	REF
If neither a link nor a session is specified for the prime or alternate node id in its definition, message 1HA4I is issued also. If no prime nor alternate route exists and the node is not defined as having either a session or link connection with our node, the entire node entry is invalidated.			
Each node entry in the new network definition table is examined for a SNA type node. If so, the APPLID is checked if another node has the same APPLID. If so, message 1RE3I APPLID FOR NODE nnnnnnnn ALREADY DEFINED IN NDT is issued anf the APPLID is invalidated.			$GAM
9 The address of the new definition table is stored in the PNET control block and the own node name is saved in the disk management block for later convience.			

This page was left blank intentionally.

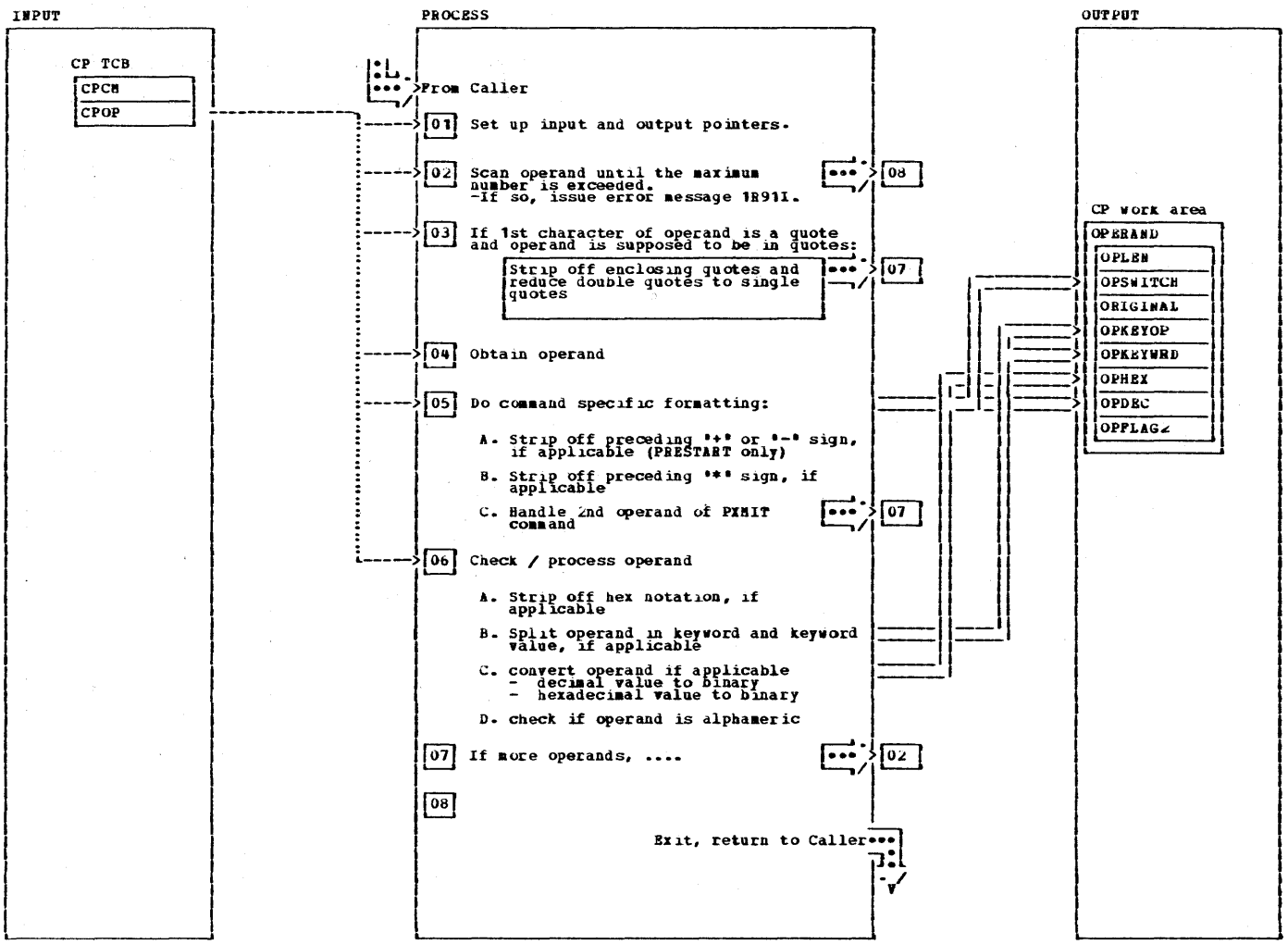
IPW$$CM - Command Processor Rootphase



NOTES	MODULE	LABEL	REF
1 To keep the permanent command processor working even if there is no storage available, the first page of the pageable area is used as a work area. If this is a temporary command processor task, a temporary CP work area is acquired.		CPR010	$RSV
3A The command code table CMNDTAB contains all valid VSE/POWER commands and the addresses to the appropriate command processor phases. While scanning thru this table, a check is made to verify that the entered command in general is a VSE/POWER command and is supported by the actual VSE/POWER environment. Following error message is issued: 1R98I cccccccc INVALID VSE/POWER COMMAND.			
3A Note: the command code can either be in long or short format. If a long format was entered, the preceding 'P' was stripped off by the attention routine or $ICP routine.			
4 A check is also made to verify that the command is acceptable during the shut-down period of VSE/POWER and is allowed for a remote operator. If such a condition is not true, appropriate error messages are issued and the incorrect command is rejected.		CPR100	

NOTES	MODULE	LABEL	REF
4 Following error messages may be issued: 1R97I cccccccc COMMAND INVALID DURING SHUTDOWN PERIOD 1R85I cccccccc COMMAND NOT ALLOWED FOR REMOTE OPERATOR			$GAM
5 The FORMAT subroutine picks up the command in the free format area of the TCB, formats the operands, converts if necessary, does some simple syntax checking and puts the formatted operands into the fixed format area (see also OPERAND DSECT layout). After return from FORMAT subroutine register 15 indicates whether formatting was successful.		CPR200	
6 The entry point address of the appropriate command processor phase is located in the command code table CMNDTAB. The command to be processed is at that time stored in the fixed-format area in the CP work area and may now be processed by the invoked command processor phase.		CPR900	
7A When an ECB address was supplied, which is the case for a temporary command processor, the ECB is posted.			
7B If the temporary command processor is active, the CP work area is released and the task detached.			$RLV
7B			$DET
7C			$WFI

FORMAT - Subroutine



NOTES	MODULE	LABEL	REF
This subroutine formats the operands of a VSE/POWER command. The positional operand is collected from the free-format area of the TCB and formatted into the fixed- format area of the CP work area. This fixed-format area is passed to the various command processor phases in order to process the command. The operand contents are, if applicable, converted into binary values. The keyword operands are split into keyword and keyword value. If the operand is surrounded by a hex notation, this is stripped off, also + and - signs and * are stripped off. Flags are set according to the characteristics of the operands.		FRM000	
Following messages may be issued: 1R91I command code TOO MANY OPERANDS, COMMAND REJECTED 1R81I command code OPERAND TOO LONG OR NO CLOSING QUOTE 1R81I command code MESSAGE/OPERAND DOES NOT START WITH QUOTE 1R52I command code LAST OPERAND INVALID			$GAM
After return from this subroutine, register 15 contains the return code: 0 - valid formatting 4 - error detected during formatting			

NOTES	MODULE	LABEL	REF
4 The remaining operand area (free format) is scanned for a delimiter (blank or comma). If not found or the length exceeds the maximum limit, the operand length is set to 255 to indicate the subsequent command procesing routine a too long operand.			

MSG - Subroutine

NOTES	MODULE	LABEL	REF
The command processor root phase IPW$$CM contains the code of this subroutine. Its entry point address is stored in the CP work area, thus making it available to all other command processor phases. This subroutine writes a message to the central operator, places a message into the remote message queue or sends a message in nodal message record format to another node. If some value must be added to the message text, a flag SWFLAG3 must have been set to indicate that register 5 contains the address of that TCB from which this value is to be fetched. Following functions are provided by this subroutine :			

NOTES	MODULE	LABEL	REF
- Write message to central operator		MSG000	$WTO
- Write message to central operator and wait for his reply		MSG020	$WTR
- Put message into remote (RJE) message queue		MSG030	$RMS
- Handle AUTOSTART messages.			$WPC
- Queue message destined for another node to appropriate node control block (NCB).		MSG060	$ICS

RELTOABS - Subroutine

NOTES	MODULE	LABEL	REF
This subroutine converts the relative queue record address passed in register 1 to an absolute address, which is stored in an eigth-byte receiving field addressed by register 2. When the queue file resides on an FBA device, it is not necessary to convert the relative queue address, because all I/O operations are performed with relative block number only. After return from the subroutine the format of the eight byte receiving field is as follows: - FBA: '0_RRBBBB' - CKD: '0__CCHHR' where: R - record number HH - track number CC - cylinder number B - FBA block number		REL000	

NOTES	MODULE	LABEL	REF
The record number is computed by dividing the relative disk address by the number of records on a track and adding one. (The one is added because the first record is used internally). The absolute track (head) number is equal the relative track number plus the starting track number. The absolute cylinder number equals the relative cylinder number plus starting cylinder number.			

VERCAUTH - Subroutine

NOTES	MODULE	LABEL	REF
This subroutine checks is the issuer of the command is authorized or not. If the issuer has not enough authorization, message IR85I cccccccc COMMAND NOT ALLOWED FOR REMOTE OPERATOR or IRA7I cccccccc COMMAND NOT ALLOWED ON NODE nnnnnnnn is issued depending on the issuer of the command. The routine is called by means of the IPW$VCA macro instruction. The macro expansion has set up register 0 with a function code, describingthe command to be executed. The function code is used as an index into appropriate tables which define for each command the authorization level required.			$GAM

NOTES	MODULE	LABEL	REF
The subroutine returns with a displacement of 0 when the issuer is not authorized or with a displacement of 4 when issuer is authorized.			

TCBSCAN - Subroutine

NOTES	MODULE	LABEL	REF
This subroutine scans the TCB chain to locate a TCB that meets 1 - 4 criteria set up by the calling routine. Criteria might be any of the following in any combination: - task id - device address - remote id (binary) - remote id (decimal). A criterion is met if the argument field matches the corresponding field in the TCB or the argument field contains zero.			

NOTES	MODULE	LABEL	REF
The field CPWTCBPT will contain the address of the TCB which meets the criteria or is zero if the scan was not successful. When continuation of the scan is requested by the calling routine, CPWTCBPT is assumed to contain the address of the TCB where the previous scan stopped. CPWTAFLG must be set if scanning at the next TCB in chain is requested.		TCBSC00	

ATTACH - Subroutine

NOTES	MODULE	LABEL	REF
This subroutine attaches a new VSE/POWER task. The task may be: - physical reader/writer task - execution reader task - RJE/BSC line manager task - RJE/SNA manager task - save account task The TCB has been previously set up by the calling routine in the dummy TCB located in the CP work area.			$ATT

NOTES	MODULE	LABEL	REF
The spool entries in the partition control block associated with the execution reader task to be attached are updated with the new TCB address. For a save-account task, the TCB address is stored in the ACB.			

ASSIGN - Subroutine

NOTES	MODULE	LABEL	REF
This subroutine updates the LUB/PUB tables of the VSE/POWER partition. Following actions are performed: 1. Locate the physical unit block for a given physical device and establish partition ownership 2. Validate if device is supported by VSE/POWER 3. Locate a free logical unit block within the LUB table of VSE/POWER partition and assign it to the given physical device. On entry to the subroutine the DOS/VS system is seized for the duration of the function by means of a SVC 42 and released on exit from the subroutine.		ASSGN00	

NOTES	MODULE	LABEL	REF
Following messages will be issued: 1R58I cccccccc DEVICE cuu IS DOWN or 1R58I cccccccc DEVICE cuu IN USE or 1R73I cccccccc INVALID DEVICE TYPE FOR ..... or 1R95I cccccccc LINE cuu NOT SUPPORTED or 1R64I cccccccc SYSLST LUB NOT AVAILABLE or 1R64I cccccccc NO FREE LUB AVAILABLE	COMREG SYSCOM		$GAM

UNASSGN - Subroutine

NOTES	MODULE	LABEL	REF
This subroutine resets LUB assignment made for a task which can not be attached due to a severe error. On entry of the subroutine register 3 must point 6 bytes before the LUB identifier (assuming CCB).		UNASN00	$ULP

NOTES	MODULE	LABEL	REF

INVDEV - Subroutine

NOTES	MODULE	LABEL	REF
This subroutine checks if the device obtained from the PUB is consistent with the task. If the device is not supported by VSE/POWER or invalid, a flag is set in SWFLAG3. On entry field 'CPWASDEV' contains the type of task and 'CPWASDTY' the PUB device type.		VDEV00	

NOTES	MODULE	LABEL	REF
Following device type categories are supported: - tape devices - reader devices - punch devices - printer devices - diskette devices - RJE control units.			

QRLNSPCT - Subroutine

NOTES	MODULE	LABEL	REF
This subroutine determines if the 1st-in-set queue record of a queue set meets the criteria set up by the calling routine. If so, the queue set is marked eligible for modifying, deleting, releasing or holding. Arguments may be: - class - jobname and jobnumber - generic jobname - binary RJE user id - local user id - password - current disposition, priority, system id - current destination.		QRLU00	

NOTES	MODULE	LABEL	REF
A criterion is met if the argument field matches the corresponding field in the queue record or the argument field contains zero. Upon exit from the subroutine register 15 contains following return code: 0 - queue set meets criteria 4 - queue set does not meet criteria.			

BINTODEC - Subroutine

NOTES	MODULE	LABEL	REF
This subroutine converts a binary number into printable decimal format. Upon exit from this routine, CONVDEC contains the decimal number in printable format, left-justified.		CVD000	

NOTES	MODULE	LABEL	REF

VQUEUEID - Subroutine

NOTES	MODULE	LABEL	REF
This subroutine checks if the operand addressed by register 4 is a valid queue identifier. Valid queue identifiers are: - LST - PRT - RDR - PUN - XMT The field CLASSPTR will contain the address of the class table associated to the specified queue or will be zero when the operand is not a valid queue identifier.		VQID00	

NOTES	MODULE	LABEL	REF
field 'CLASSPC3' contains x'80' - for RDR x'40' - for LST or PRT x'20' - for PUN x'10' - for XMT or an unpredictable value if none of above queue identifiers is specified. Field 'CLASSLC' contains the maximum number of class entries in that particular queue.			

VTASKID - subroutine

NOTES	MODULE	LABEL	REF
This subroutine checks if the operand addressed by register 4 is a valid task identifier. Valid task identifiers are : - LST - PRT - RDR - PUN After return from this subroutine register 15 contains the return code: 0 - valid task identier 4 - invalid task identifier		VTID00	

NOTES	MODULE	LABEL	REF

VPARTID - Subroutine

NOTES	MODULE	LABEL	REF
This subroutine checks if the operand addressed by register 4 is a valid DOS/VSE partition identifier. This is done by scanning the PIB table for a matching partition identifier.	SYSCOM		

NOTES	MODULE	LABEL	REF
After return from this routine OTHPIBPT will contain the address of the PIB for the partition concerned or will be zero when the operand is not a valid partition identifier.	COMREG	PIBSC00	

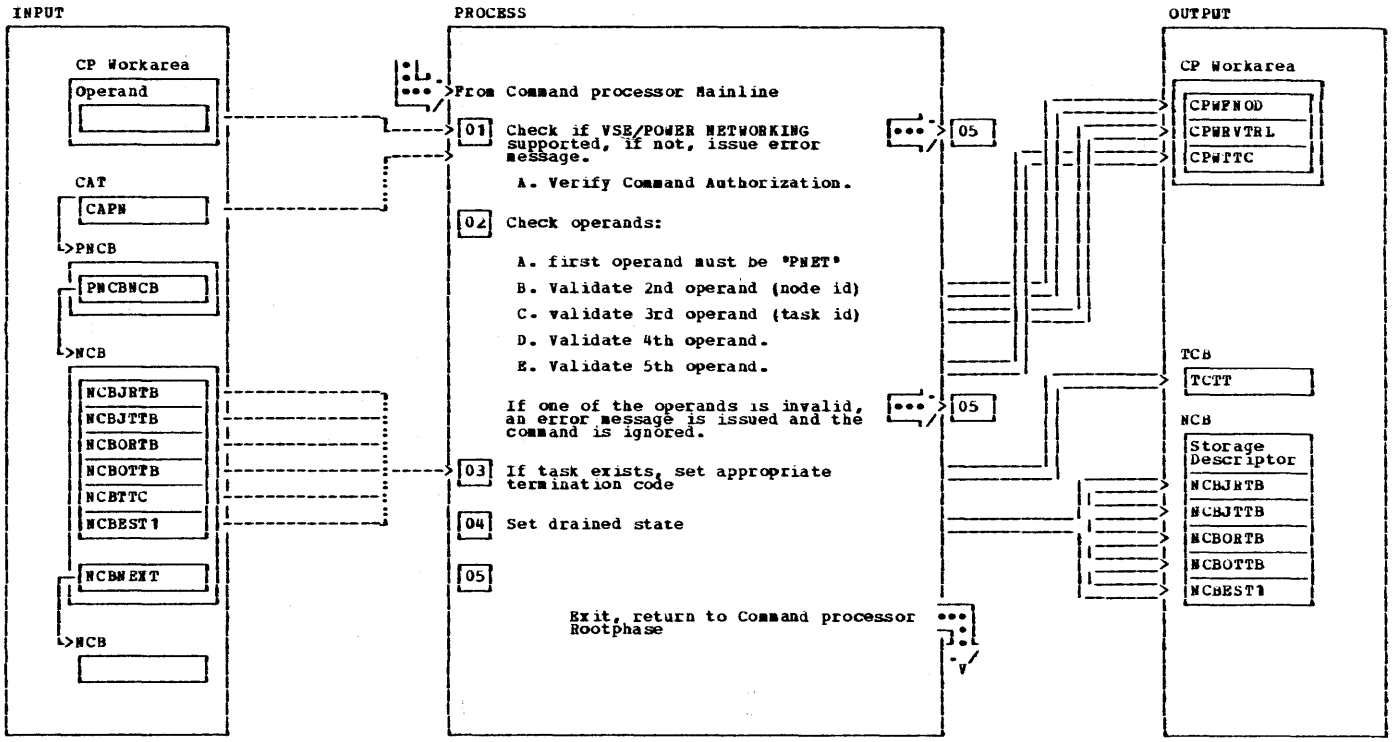
VERKEYOP - Subroutine

NOTES	MODULE	LABEL	REF
This subroutine checks if the operand addressed by register 4, is one of the following keywords and that the keyword value is properly specified. Valid keywords are: - CDEST - CDISP - CPRI - CSYSID		VQKW00	
If the operand is none of above keywords, message 1R52I cccccccc OPERAND ** NOT SPECIFIED AS VALID KEYWORD is issued,			$GAM
If the keyword value is wrong, message 1R52I cccccccc INVALID SPECIFICATION FOR KEYWORD .... is issued			$GAM
The subroutine returns with a displacement of 0, when an error has been detected (either keyword or keyword value wrong) or with a displacement of 4 when the operand is correct.			

CLASS - Subroutine

NOTES	MODULE	LABEL	REF
This subroutine checks if the class(es) specified in the argument list in the CP work area is (are) valid. If so, this subroutine places in the TCB, addressed by the field CPWTCBPT, pointers to master class table entries. Input arguments are : CPWCLAS - class(es) CPWCLTI - task type, either RDR,LST or PUN CPWCLDF - default class: X"00" for physical tasks; 0 for BG partition; 1 for F1 partition; etc. CPWCL#C - max. number of classes allowed		CLSS00	
Each character of field "CPWCLAS" is checked until a blank character is found. If the maximum number of characters allowed are processed, message 1R87I cccccccc TOO MANY CLASSES, FIRST n PROCESSED is issued.			$GAM
If the character is invalid according to the task type, message 1R54I cccccccc CLASS x INVALID is issued.			$GAM
After return from this routine, register 15 contains the return code: 0 - valid class specification 4 - invalid class specification	COMREG		
The table below shows which classes are valid for a specific task: Task type: Valid classes: LST: A - Z PUN: A - Z RDR: A - Z, 0 up to highest DOS/VSE-supported partition number RDR(ex.RDR): A - Z, corresponding partition id (0 for BG)			

IPW$CN - PDRAIN Processing

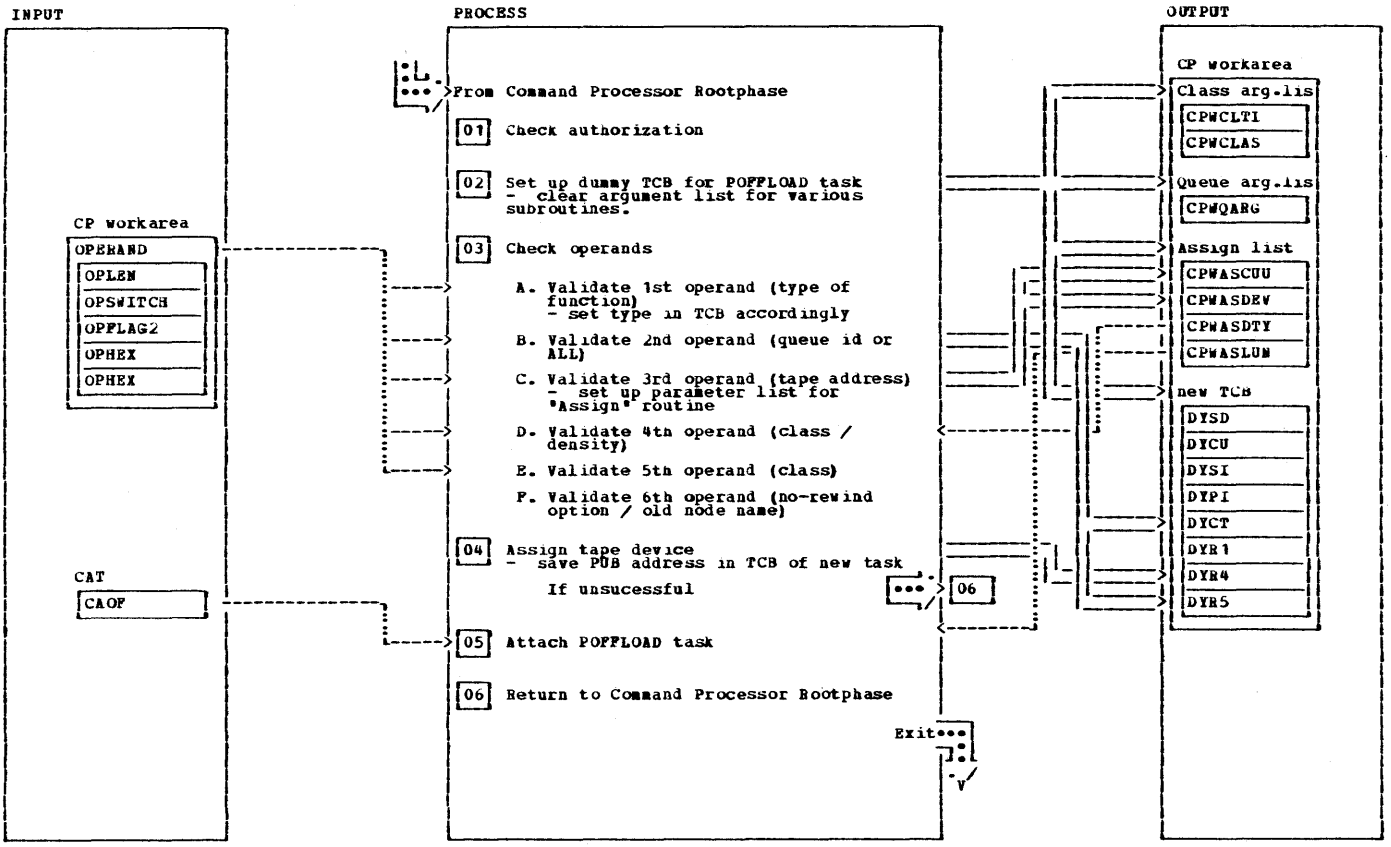


NOTES	MODULE	LABEL	REF
1 A first a check is made if VSE/POWER networking is supported. If not, following error message is issued: 1RA3I VSE/POWER NETWORKING NOT SUPPORTED A 2nd check verifies the Authorization of the issuer.		PDRNT00	
2A The first operand must be 'PNET'. If not, error message 1R52I OPERAND ## MISSING OR INVALID is issued.			
2B The second operand specifies the node id for the task to be drained. The node id must be alphameric, first character alphabetic and up to a max. length of eigth bytes. If the operand is errornous, message 1R52I OPERAND ## MISSING OR INVALID is issued.			
2B The NCB chain is scanned to locate the appropriate NCB. If no NCB is found, indicating that no connection is established or even that the node is unknown, message 1RB0I NO CONNECTION ESTABLISHED TO NODE nnnnnnnn is issued.			
2C The 3rd operand is mandatory and specifies the task id of the task to be drained. Valid specifications are 'TR1 - TR7' for transmitter tasks or 'RV1 - RV7' for receiver tasks. 'TR*' or 'RV*' are also valid. If no valid task id is specified, message 1R52I OPERAND ## MISSING OR INVALID is issued.			

NOTES	MODULE	LABEL	REF
2D The 4th operand specifies what the task to be drained is supposed to do, either to process jobs or output (LST or PUN). Valid specification is 'OUT' or 'JOB'. If none of above specifications, message 1R52I OPERAND ## MISSING OR INVALID is issued.			
2E The 5th operand, if not omitted, must be 'EOJ'. It indicates that the appropriate task should be drained at end of job rather than immediately. Default is immediately. If operand is not 'EOJ' nor omitted, message 1R52I OPERAND ## MISSING OR INVALID is issued.			
3 The Status Byte in NCB task entry is tested wether a task exists. If so the termination code (either 'S' or 'E') is set in the TCB.		PDRNT80	
4 The drained flag is turned on in the appropriate task table contained in the NCB.			

This page was left blank intentionally.

IPW$$CO - POFFLOAD Processor

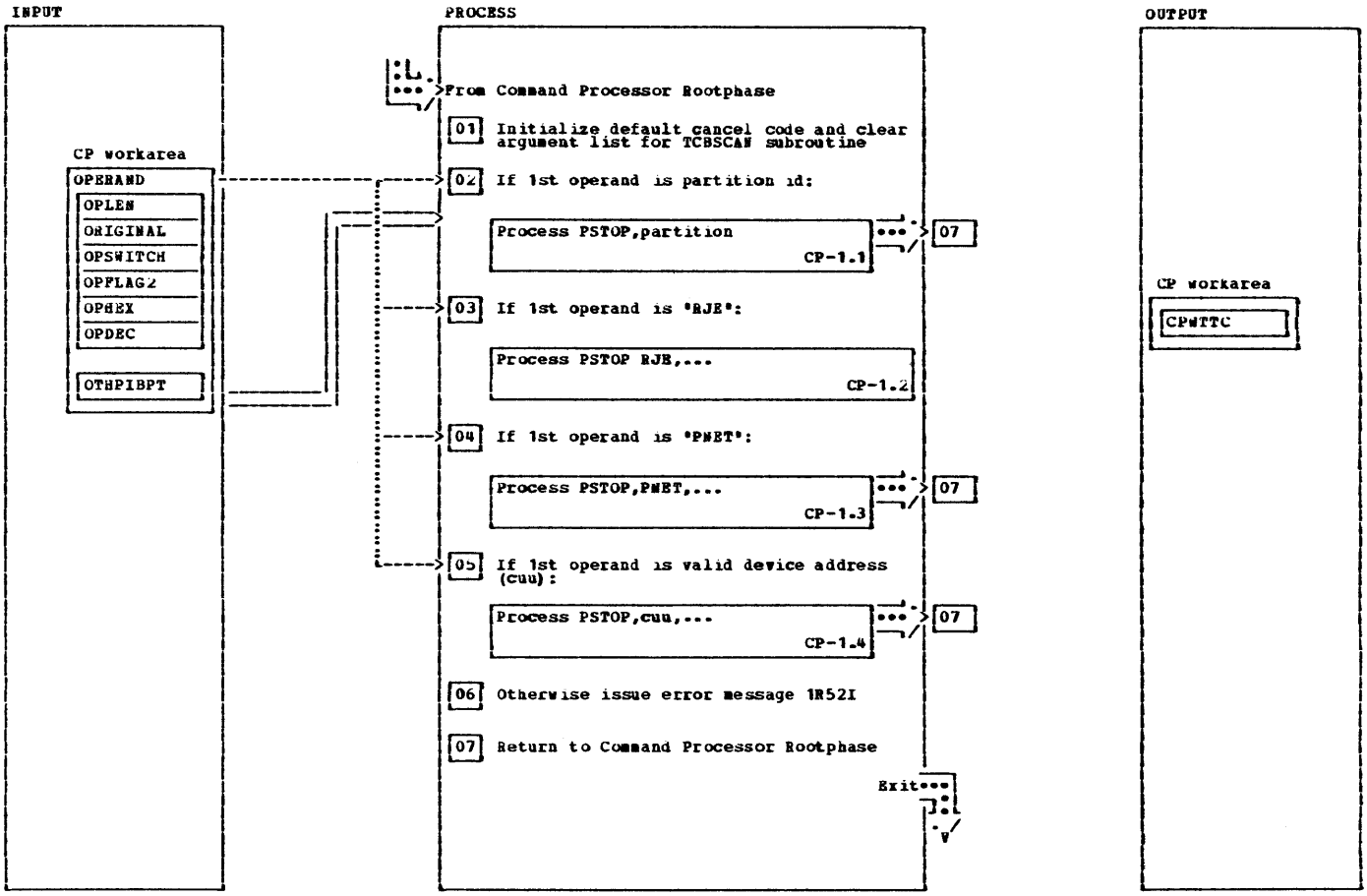


IPW$$CO - POFFLOAD Processor

NOTES	MODULE	LABEL	REF
1 The command authorization subroutine, located in rootphase of the command processor, is called to check if the issuer of the command has enough autorization to do it. If not, the command is rejected.		POFF000	$VCA
3A The first operand is mandatory and is either 'SAVE' or 'LOAD'. If none of above is specified, message 1R52I cccccccc OPERAND ## MISSING OR INVALID is issued.		POFF100	$GAM
3B The 2nd operand specifies the queue to be saved or re-loaded. Valid specifications are 'RDR', 'LST', 'PUN', 'XMT' or 'ALL'. In the last case all queues are processed. The 'Verify queue id' subroutine is called for this purpose. If none of above is specified, message 1R52I cccccccc INVALID QUEUE SPECIFICATION is issued.		POFF200	$GAM
3C The 3rd operand specifies the address of the tape device to be used. If invalid, message 1R52I cccccccc OPERAND ## MISSING OR INVALID is issued.		POFF300	$GAM
3D For the 'LOAD' function the operand specifies the class which will be given to all queue sets being re-loaded. The operand is optional and if omitted, the queue sets keep their original class. If specified, the 'CLASS' subroutine is called to verify the class specification. The command is rejected when the class specification is invalid.		POFF400	
3D The operand defines for the 'SAVE' function the density of the tape device. If the operand is omitted, the default density is taken, otherwise the specified density is checked according to the device type. To check for valid tape address and density the PUB is scanned for matching specifications. If the specified device (cuu) is not defined in the DOS/VSE system, message 1R58I cccccccc DEVICE cuu IS NOT KNOWN is issued. If the device is not a tape device, message 1R74I cccccccc INVALID DEVICE SPECIFICATION is issued. if a wrong density is specified, message 1R53I cccccccc INVALID DENSITY is issued.			$GAM

NOTES	MODULE	LABEL	REF
3E The operand defines for the 'SAVE' function the class(es) of the appropriate queue to be saved. Valid specifications are 'A - Z', '0 - 9' or '*' in which case all classes are processed. The operand is invalid if either all queues or the XMIT queue are to be saved. The operand is optional and if omitted, the default class is 'A'. If specified, the 'CLASS' subroutine is called to verify the class specification. The command is rejected when the class specification is invalid.		POFF500	
3E In case of invalid class specification subroutine CLASS has already issued error message, so a branch is taken to processor exit			
3E The operand defines for the 'LOAD' function the no rewind option. If 'NOREW' is specified, the poffload task does not rewind when all queue entries are saved nor rewinds to the tape load point at the beginning.			
3F The operand defines for the 'SAVE' function the no rewind option. If 'NOREW' is specified, the poffload task does not rewind when all queue entries are saved nor rewinds to the tape load point at the beginning. The operand defines for the 'LOAD' function, the old node name. If a node name is specified, the name is saved in the new TCB.			
4 In case of unsuccessful assignment subroutine ASSIGN has already issued error message, so a branch is taken to processor exit		POASS00	
5 Task identifiers are 'WLST' for SAVE function and 'RRDR' for LOAD function. Tape address, density and class paramters are stored into TCB			

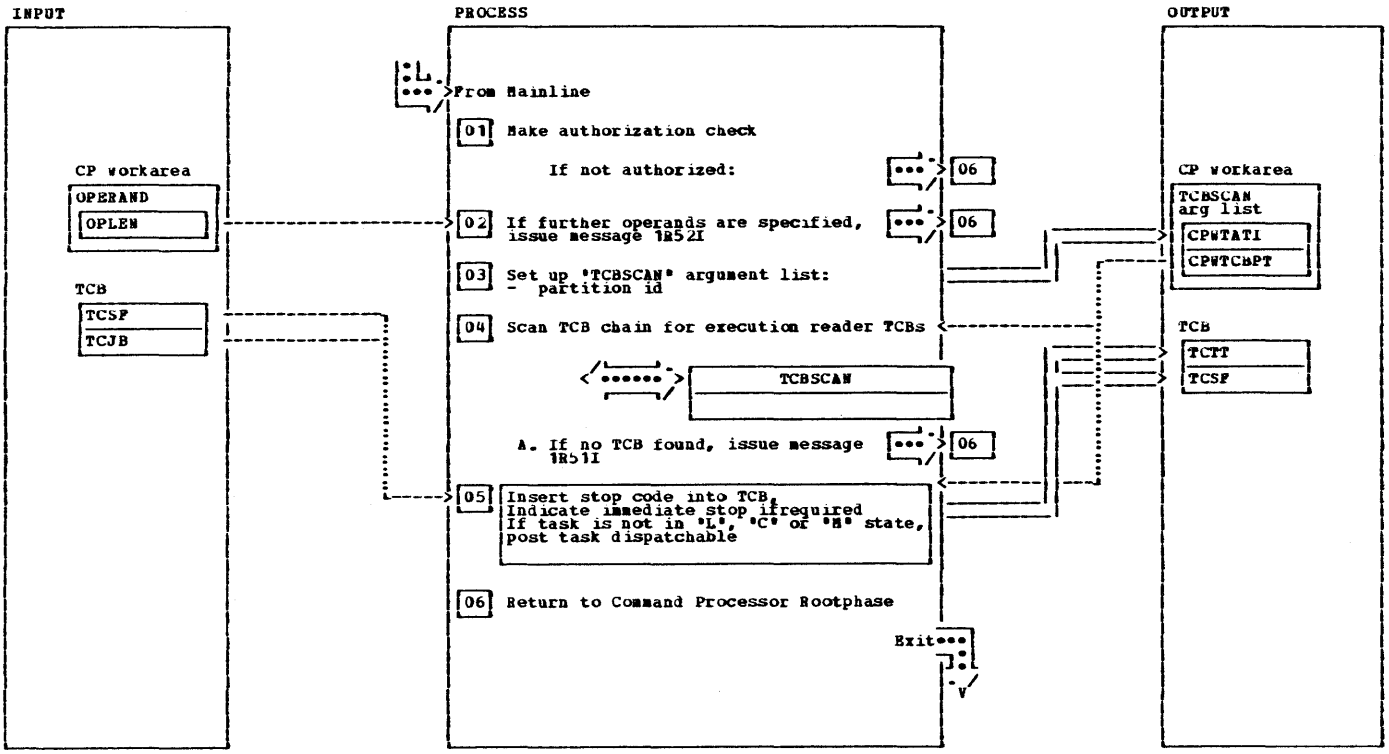
IPW$$CP - PSTOP Processor



NOTES	MODULE	LABEL	REF
The PSTOP command has the following format either PSTOP uraddr<,EOJ> or PSTOP uraddr<,RESTART> or PSTOP partition or PSTOP lineaddr<,EOJ> or PSTOP RJE,SNA,<luname><,EOJ> or PSTOP PNET,nodeid<,EOJ> where is: uraddr : Specifies a device address in the form cuu or x'cuu' partition : Specifies a valid partition identifier lineaddr : Specifies a line address in the form cuu or x'cuu' luname : Specifies a name of a logical unit nodeid : name of node EOJ : de-activate when transmission complete			

NOTES	MODULE	LABEL	REF
I general the task control block, specified by the first operand, is provided with the cancek code 'E' or 'S'. Note: * ..STOP RJE commands are not supposed to be processed by this processor.			
2 The 'Verify partition Id' subroutine is called to check if the 1st operand is a valid partition id, known in the DOS/VSE system. If so, the subroutine returns in field 'CPWPIBPT' the address of the PIB of the partition concerned. Otherwise the field contains zeros.			
5 That can be either i line address or a unit record device address.			
6 Following message will be issued: 1R52I cccccccc OPERAND ## MISSING OR INVALID			$GAM

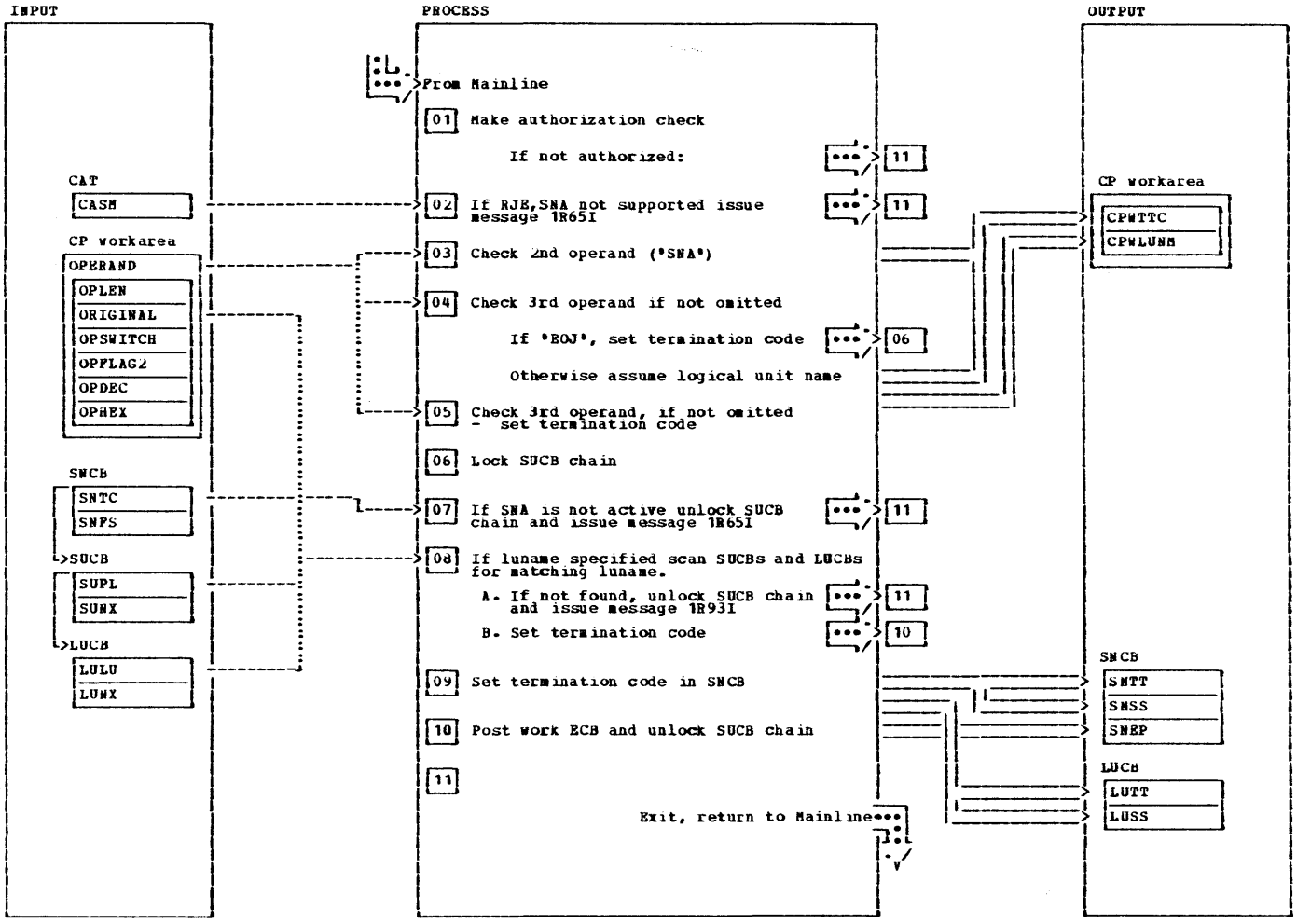
Process PSTOP of partition



NOTES	MODULE	LABEL	REF
1 The 'Verify command authorization' subroutine is called to check if the issuer of the command has enough authorization. If not authorized, the command is rejected.			$VCA
2 Following message will be issued: 1R52I cccccccc OPERAND ## INVALID			$GAM

NOTES	MODULE	LABEL	REF
4 The subroutine reurns in field 'CPWTCBPT' the address of the TCB macthing the criteria set up in the argument list.			
4A Following message will be issued: 1R51I cccccccc OPERAND 1 DESIGNATES NON-EXISTING TASK			$GAM

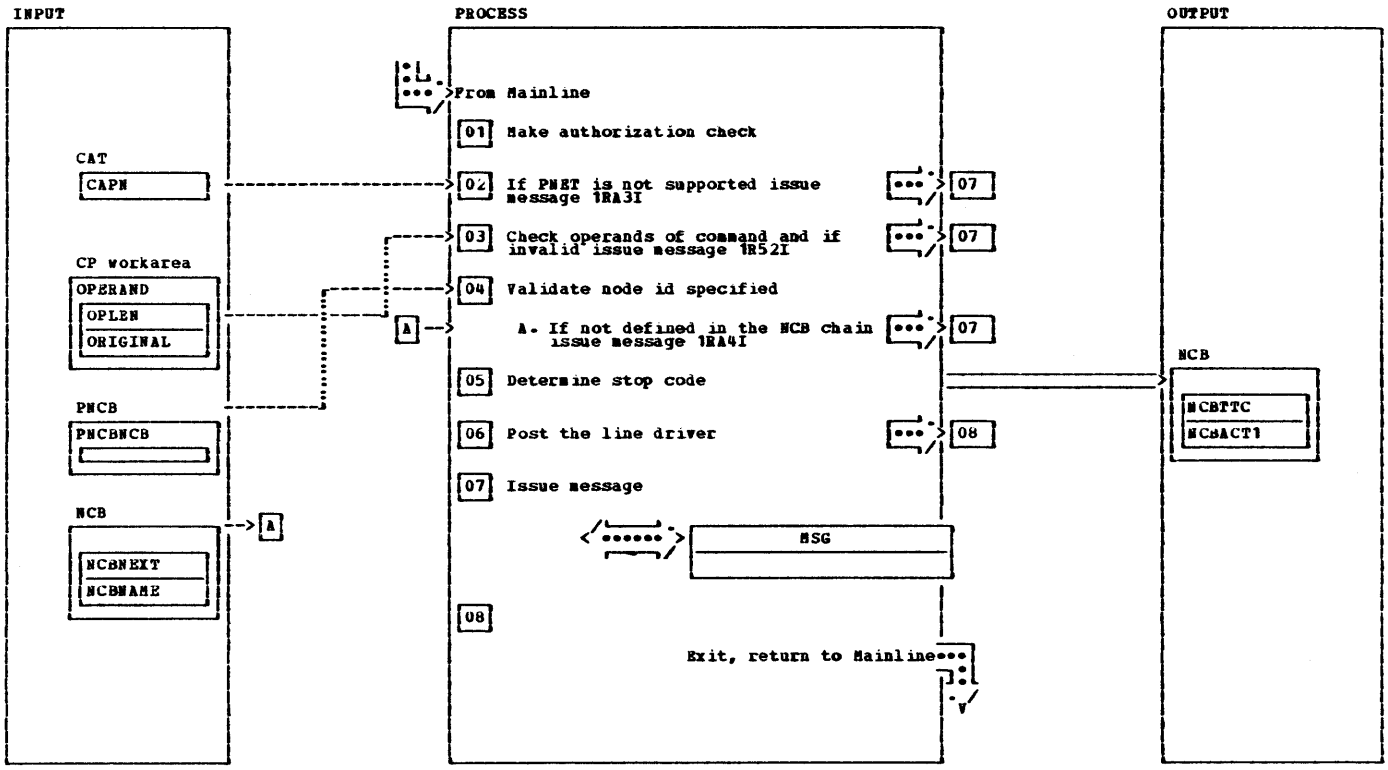
IPW$$CP - PSTOP,RJE,SNA Processor



NOTES	MODULE	LABEL	REF
1 The 'Verify command authorization' subroutine is called to check if the issuer of the command has enough authorization. If not authorized, the command is rejected.			$VCA
2 Following message will be issued: 1R65I cccccccc RJE,SNA NOT SUPPORTED			$GAM
3 The 2nd operand must be 'SNA'. If omitted or wrongly specified, message 1R52I cccccccc OPERAND ## MISSING OR INVALID is issued.			$GAM
4 A valid logical unit name is alphameric, first character alphabetic up to a length of 8 bytes. If invalid, message 1R52I cccccccc OPERAND ## MISSING OR INVALID is issued.			$GAM

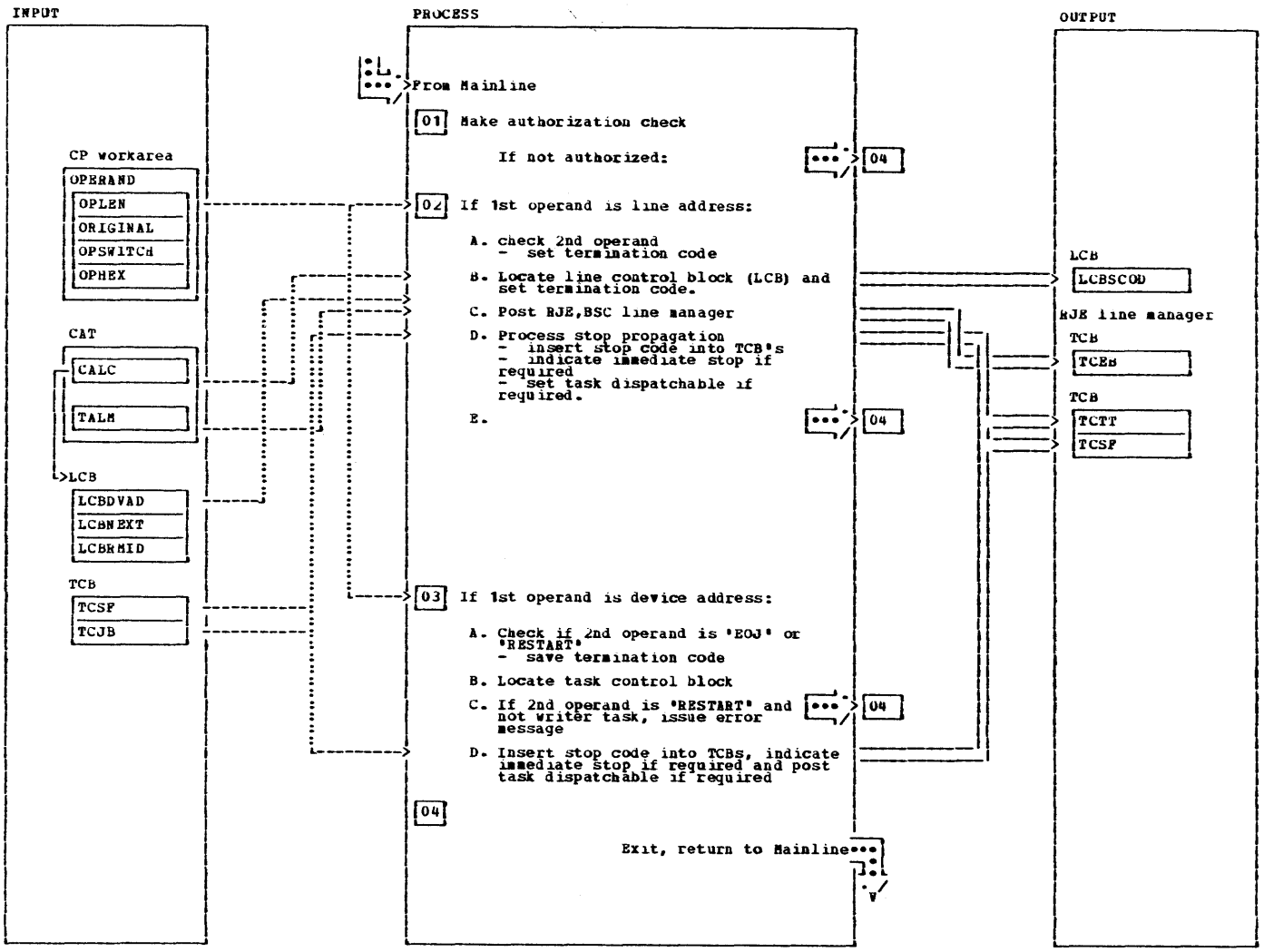
NOTES	MODULE	LABEL	REF
5 The 3rd operand must be 'EOJ'. If invalid, message 1R52I cccccccc OPERAND ## MISSING OR INVALID is issued.			$GAM
6 The SUCB chain is locked for the duration of the scan.			$HSR
7			$HLR
7 Following message will be issued: 1R65I cccccccc RJE,SNA NOT STARTED			$GAM
8A			$HLR
8A Following message will be issued: 1R93I cccccccc NO SESSION ESTABLISHED FOR LUNAME			$GAM
10			$HLR

IPW$$CP - PSTOP/PNET Processor



	NOTES	MODULE	LABEL	REF
1	Issue IPW$VCA PSTOP,PNET to check authorization			$VCA
2	Following message will be issued: 1RA3I command code VSE/POWER NETWORKING NOT SUPPORTED			
3	The only valid specification for the 1st operand is 'PNET'. If not issue message: 1R52I operand xx missing or invalid The 2nd operand specifies the node id whose line is to be stopped. It is manadatory, alphameric (first character alphabetic), length <=8. If not issue message: 1R52I operand xx missing or invalid. Scan the NCB chain to determine if node id is valid, if not issue message:			

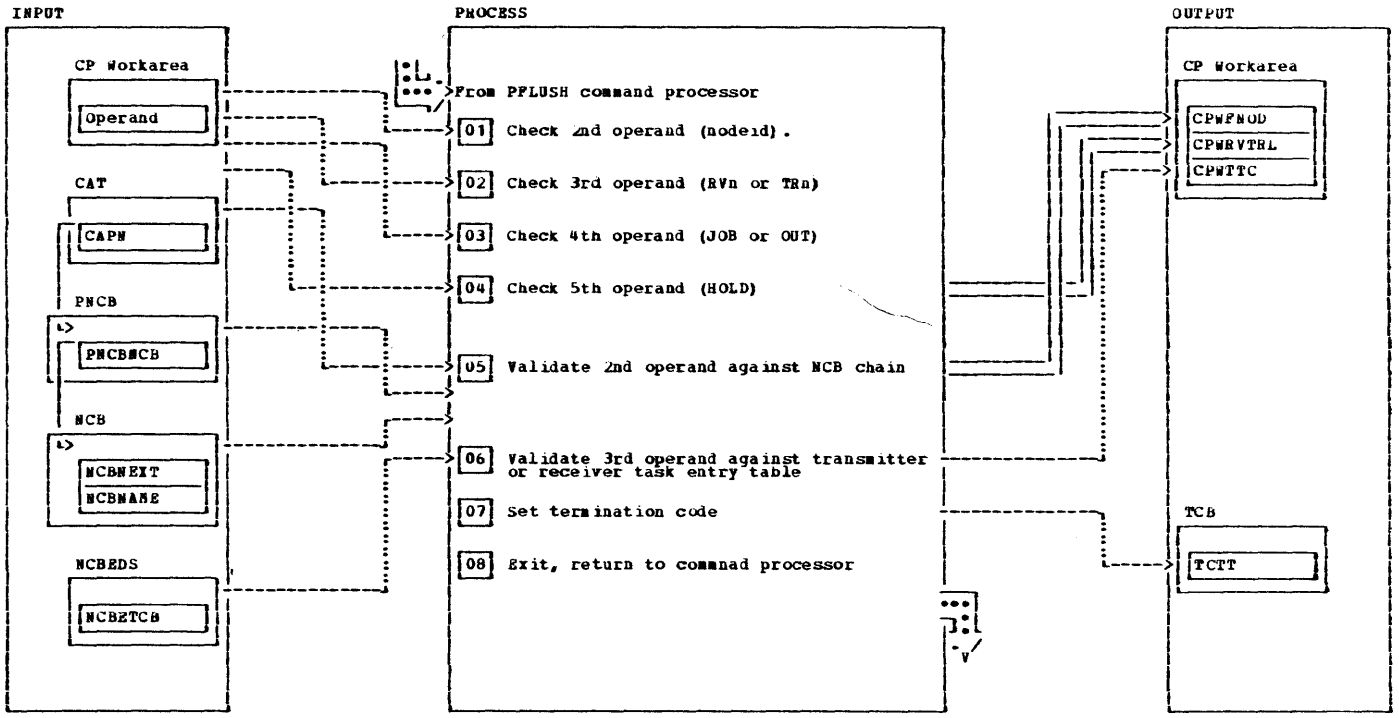
	NOTES	MODULE	LABEL	REF
				$SB
	1RA4I xxxxxxxx invalid node-id The 3rd operand is optional. Only when current transmission is complete will it be de-activated. The only valid specification is 'EOJ'. If not issue message: 1R52I operand xx missing or invalid			
5	Immediate stop flag will be set for default when 'EOJ' not specified. Immediate stop will take priority only if NCB termination type is already 'e' for eoj or if blank. Termination type of 'e' will be set in the NCB when no termination code already exists.			
6	Post the line driver so that he can propagate the stop condition			
7				$GAM
8				$RLR



	NOTES	MODULE	LABEL	REF
1	The 'Verify command authorization' subroutine is called to check if the issuer of the command has enough authorization. If not authorized, the command is rejected.			$VCA
2A	The 2nd operand, if specified, must be 'EOJ'. If not, message IR52I ccccccc OPERAND ## MISSING OR INVALID is issued.			$GAM
2D	The TCB selection list is now scanned to locate any reader list or punch task existing for the line being stopped. If so, the task is stopped by setting the stop code into the TCB and setting the task dispatchable if necessary. The 'TCBSCAN' subroutine is used for this purpose.			

	NOTES	MODULE	LABEL	REF
3A	The 2nd operand, if not omitted, can either be 'EOJ' or 'RESTART' which is only valid for a writer task. If none of above is specified, message IR52I ccccccc OPERAND ## MISSING OR INVALID is issued.			$GAM
3C	Following error is issued IR52I ccccccc OPERAND ## MISSING OR INVALID			$GAM

IPW$$CPF - PFLUSH/PNET Processor

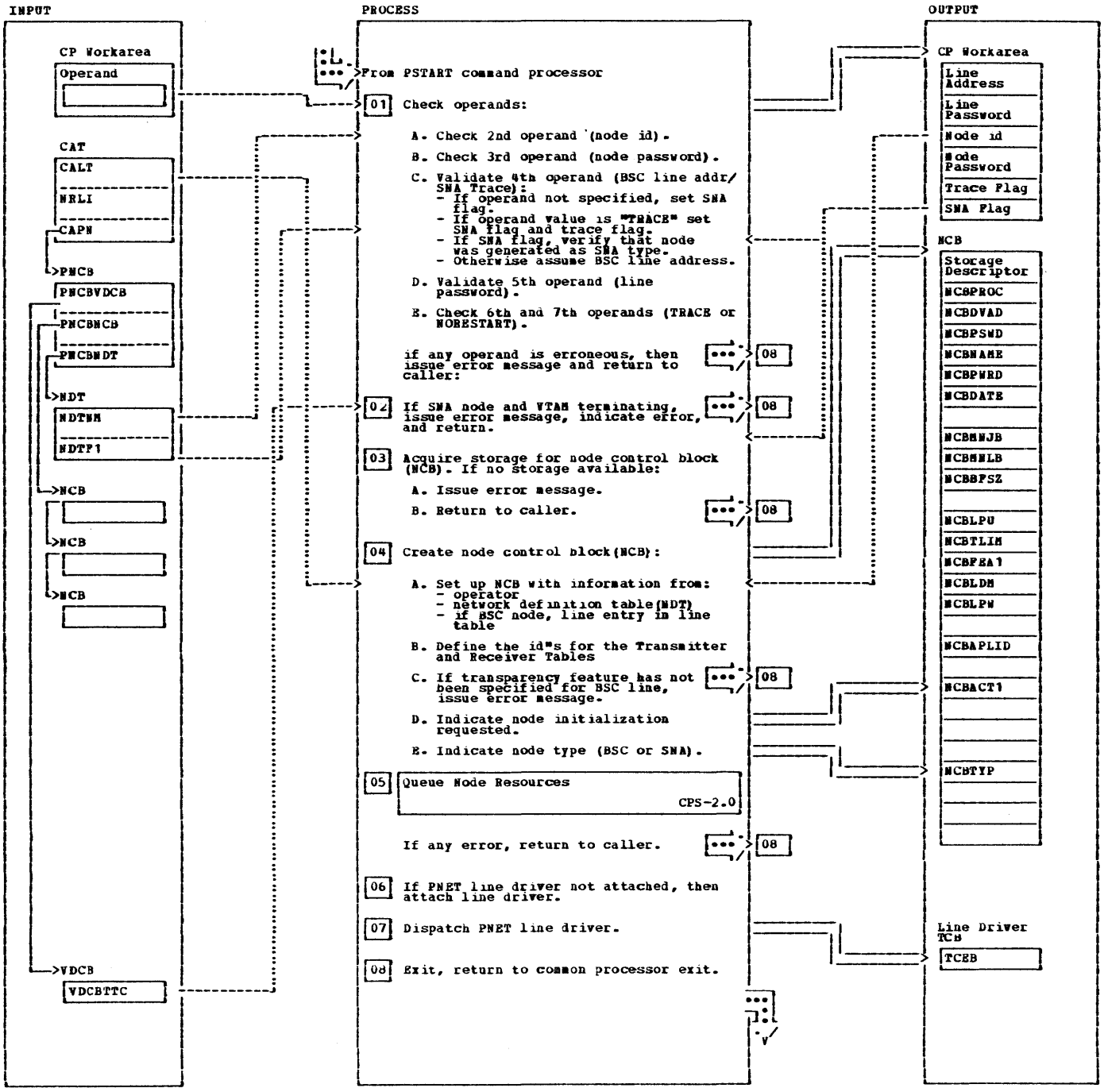


NOTES	MODULE	LABEL	REF
1 The 2nd operand specifies the node id, to which a transmitter or receiver flush will be requested. -It is mandatory -It is alphameric -length not greater than 8. If no, issue message: 1R52I operand xx missing or invalid			
2 The 3rd operand specifies whether we are flushing a transmitter or a receiver and also specifies the specific line. That is RVn is valid when n=1 to 7 -It is mandatory -The only valid specification is RVn or TRn where n=1 to 7. If not valid issue message: 1R52I operand xx missing or invalid			
3 The 4th operand specifies whether the transmitter or the receiver is a job or output receiver or transmitter. -It is mandatory -The only valid specification is 'JOB' or 'OUT' Otherwise issue error message: 1R52I operand xx missing or invalid			
4 The 5th operand specifies that the queue entry of the flushed VSE/POWER job/output is not to be deleted but placed in a HOLD state. -It is optional for a transmitter task. -It is not valid for a receiver task. -The only valid specification is 'HOLD'. Otherwise issue error message:			

NOTES	MODULE	LABEL	REF
1R52I operand xx missing or invalid			
5 Reserve the PNCB and get addressability to the NCB. Scan the NCB chain until node id match is found. If not found issue message: 1RA4I xxxxxxxx invalid node-id			$RSR
6 Index the Transmitter or Receiver Table to find task entry # match. If no TCB found issue: 1R51I non-existing task designated			
7 Set the TCB termination code equal to the following: 's' - if receiver was specified 'h' - if transmitter hold was specified 'f' - if only transmitter was specified (no hold was specified) Release the PNCB			$RLR

This page was left blank intentionally.

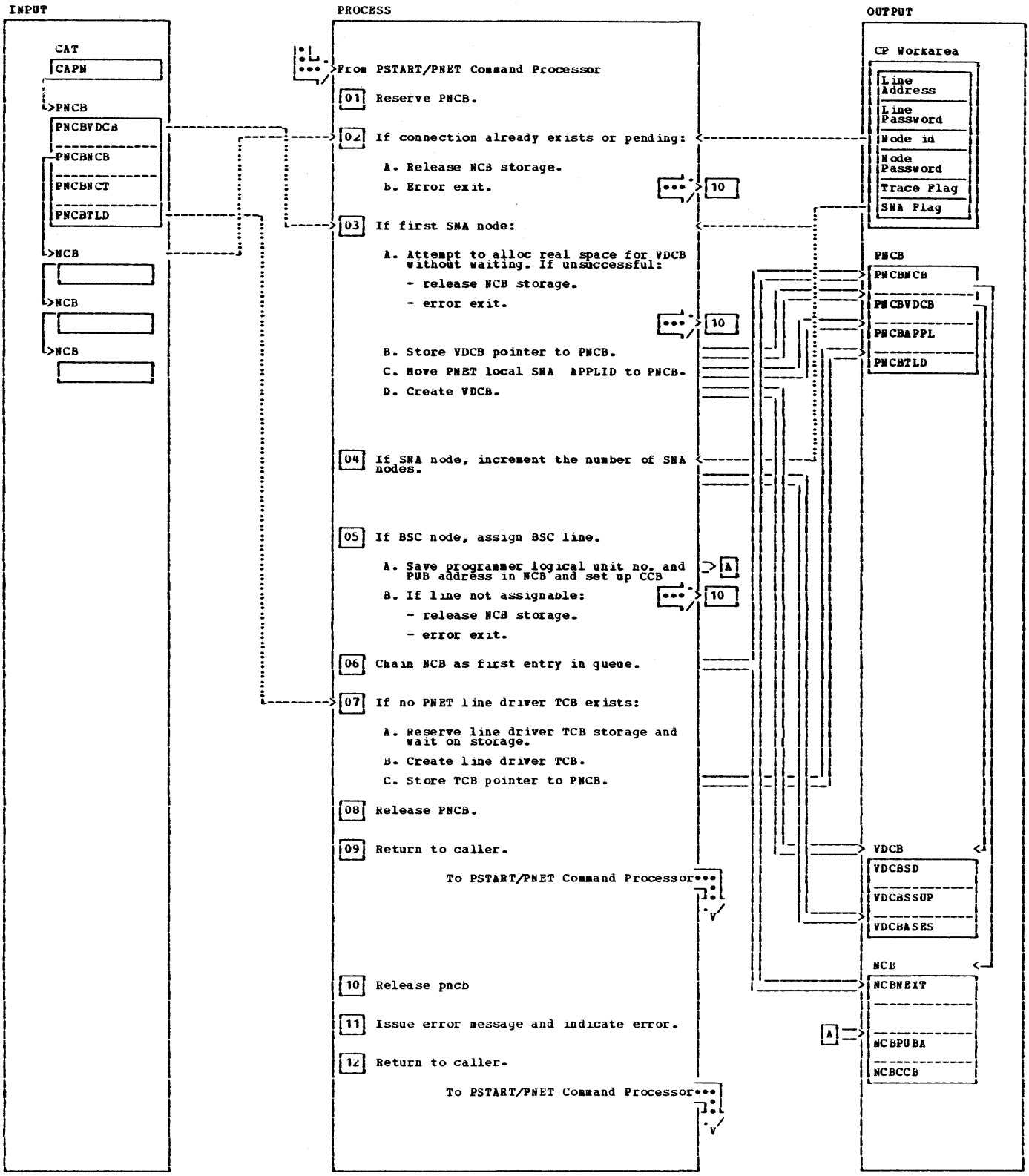
IPW$$CPS- PSTART/PNET Processor



NOTES	MODULE	LABEL	REF
1A The 2nd operand specifies the node id, to which the connection should be established. -It is mandatory -The first character must be alphameric, and the length not greater than 8. If no, issue message: 1R52I OPERAND xx MISSING OR INVALID			$GAM
The NETWORK Table is scanned to examine if node id is defined. If no, message: 1RA4I xxxxxxxx INVALID NODEID is issued. If the node id specifies the own node id (first entry in table) the same error message is issued. If the node id is not for an adjacent (link) node then the message 1RA4I is also issued. Otherwise the node id is saved in the CP Workarea.			$GAM
1B The 3rd operand is optional and specifies the node password. -Alphameric (first character alphabetic), length <= 8. If invalid, issue error message: 1R52I OPERAND xx MISSING OR INVALID Otherwise the password is saved in the CP Workarea.			$GAM
1C IF the 4th operand specifies a BSC line address: -It must be hexadecimal -It is mandatory If the line address is valid, it is moved in the CP Workarea for later usage. Otherwise issue error message: 1R74I INVALID LINE ADDRESS			$GAM
If the PSTART is in the SNA node format (no operand specified), then the node generation in the PNODE macro is checked: the flag NDTVA in the flag byte NDTF1 must be set - if not set, then issue the error message: 1RA4I INVALID NODEID xxxxxxxx			$GAM
1D The 5th operand specifies the line password: -It is optional, if omitted a blank password is assumed -It must be alphameric -Its length is not greater than 8. If so, password is saved in the CP Workarea. Otherwise message: 1R52I OPERAND xx MISSING OR INVALID is issued.			$GAM
1E For a BSC node, the last two operands can be TRACE or NR. Both are optional and can be specified in any order. For an SNA node, the TRACE operand may be specified. The seventh operand is ignored. TRACE means that all I/O events for this node are to be written into a trace area. NR means that no restart for the node has to be tried whenever the node is signed-off due to time-outs. If neither TRACE nor NR has been specified: 1R52I OPERAND xx MISSING OR INVALID is issued.			$GAM

NOTES	MODULE	LABEL	REF
2 VDCBTTC will be checked if the following bits are set: VDCBTTCV = ACF/VTAM Abend VDCBTTCE = ACF/VTAM normal shutdown If VTAM terminating, issue message: 1RD0I PSTART COMMAND IGNORED, ACF/VTAM TERMINATING			$GAM
3 Storage is reserved for the node control block. If no storage can be obtained, message: 1078I NO REAL/PFIXED STORAGE AVAILABLE is issued.			$RSW $GAM
4 The node control block is formatted: -storage descriptor -system date -task entries.			
4A The information used from the operator: -Node id -line address -password (line & node) -header descriptor -trace option The information used from the NDT: -If SNA, the APPLID is moved to the NCB -buffer size -max number of buffers for transmitters and receivers. If BSC node, the line table, specified by the PLINE macro at VSE/POWER generation time, is scanned to locate the line entry. The information used from the line table: -time out limits -line features -line password (if necessary) The line password is only moved to the NCB, when no password has been specified.			
4C When the line is not specified with the transparency feature in the PLINE macro, issue message 1R06I LINE cuu NOT TRANSPARENT			$GAM
5 The node control block is chained as first entry in the NCB chain.			
6 The line driver TCB is attached and formatted in the area contained in the CP workarea.			$ATT
7 The NETWORK Line driver is set dispatchable.			

IPW$$CPS- Queue Node Resources

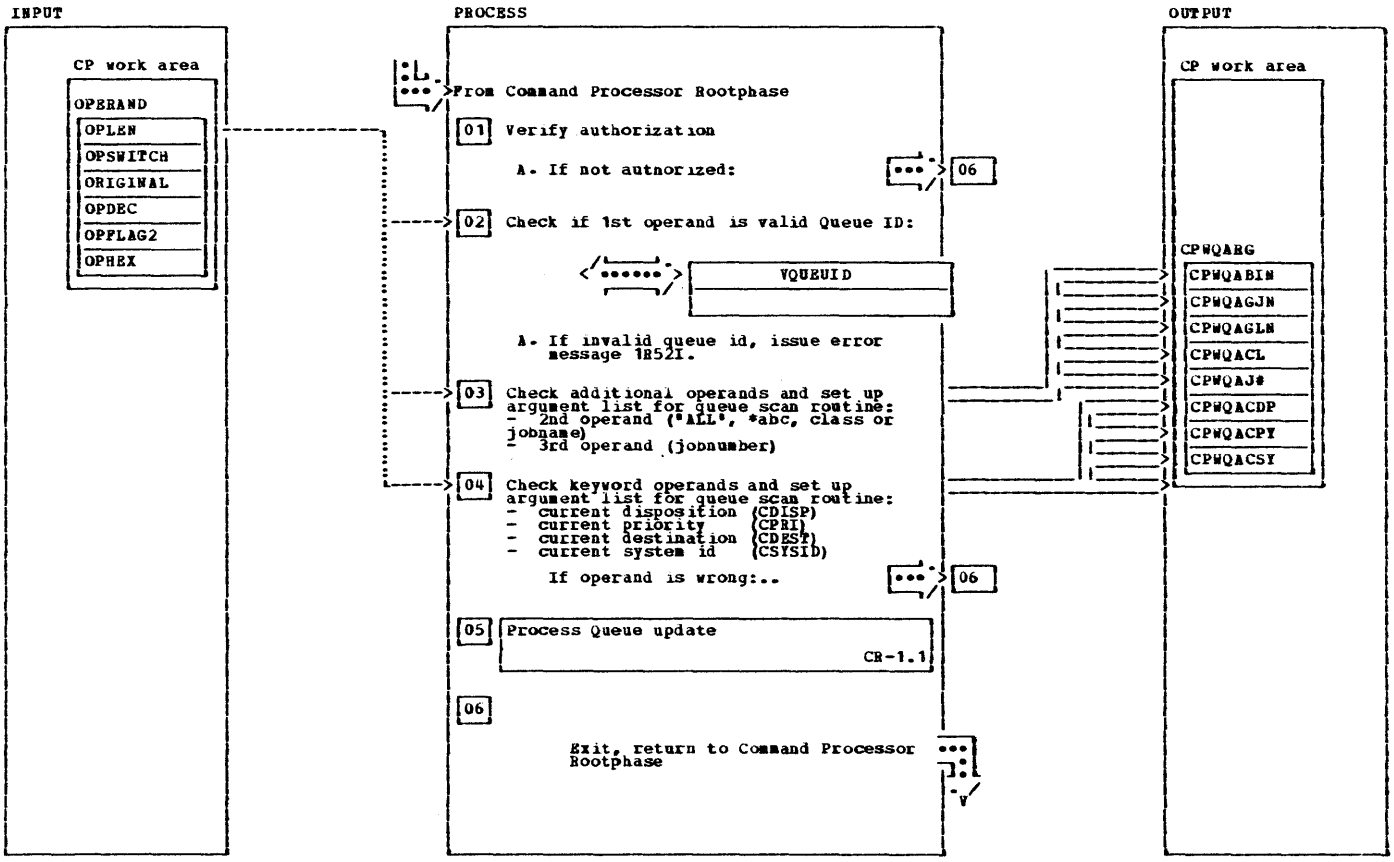


IP#$$CPS- Queue Node Resources

NOTES	MODULE	LABEL	REF
1 Lock P#CB resource.			$RSR
2A Release #CB storage.			$RLW
2A If the node is already started or pending issue message: 1RD1I PSTART #ODE nnnnnnnn ALREADY STARTED			$GAM
3A Reserve storage for VDCB.			$RSW
3A Issue message: 1078I #O REAL/PFIXED STORAGE AVAILABLE			$GAM
3D The VDCB is initialized with: - VDCBSD = storage descriptor - VDCBSSUP= SNA start up req.			

NOTES	MODULE	LABEL	REF
5 The line address is assigned, the PUB Address and the logical unit are saved in the new created #CB. If the specified cuu is not allowed, several error messages are possible (see HIPO for ASSIGN routine).	ASSIGN		
7A The P#ET line driver storage is reserved with "wait" since this occurs only with the first node, and this implies that no tasks are running which could be waiting on the P#CB resource.			
7C The address of the line driver TCB is stored in the P#CB.			
8 Unlock P#CB resource.			$RLR

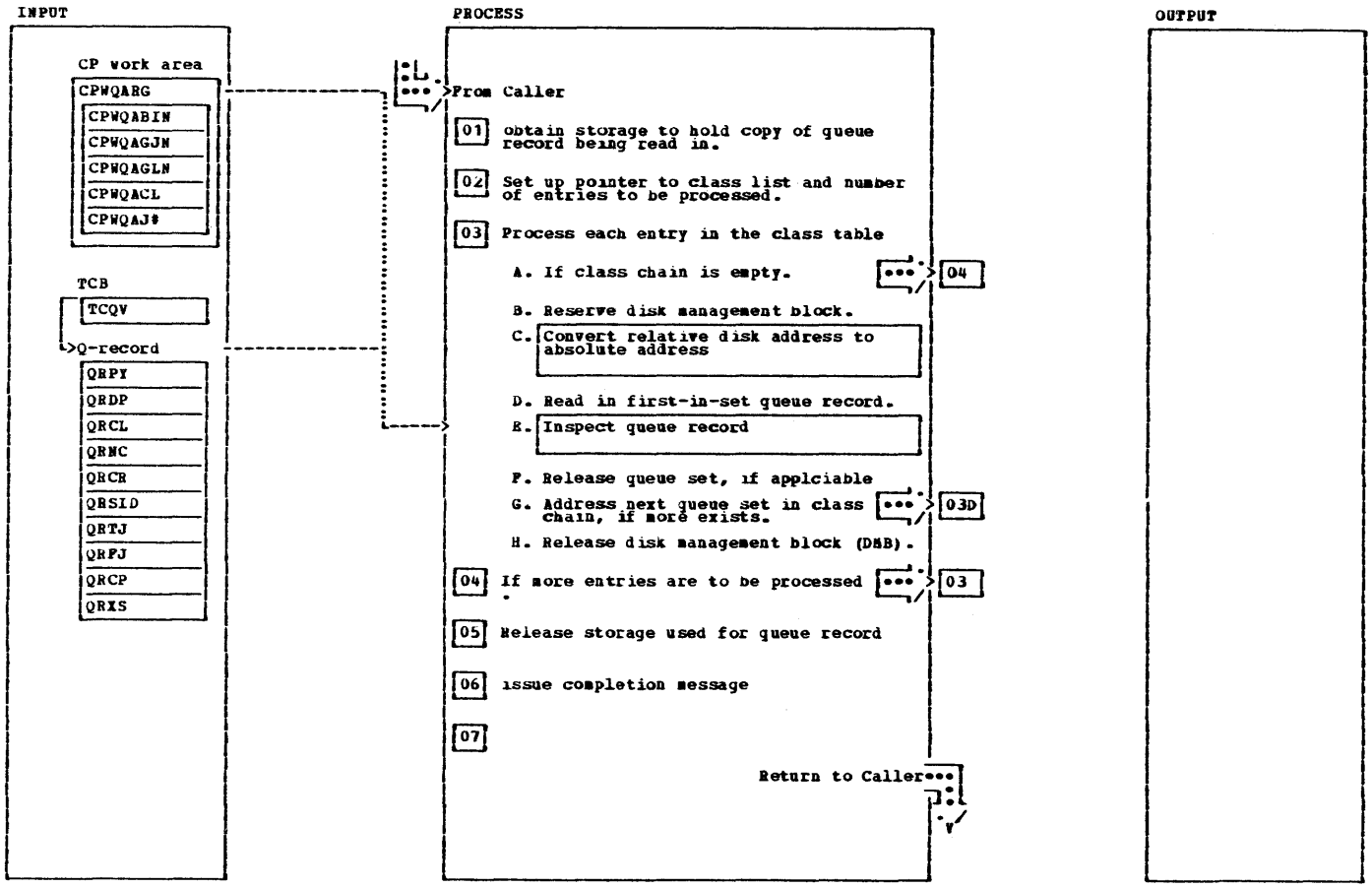
IPW$$CR - PRELEASE Processor



NOTES	MODULE	LABEL	REF
The PRELEASE command has following possible formats: PRELEASE queue,jobname<,jobnumber> <,keyword=value> PRELEASE queue,ALL <,keyword=value> PRELEASE queue,class <,keyword=value> PRELEASE queue,*abc <,keyword=value> where: queue : One of LST,RDR,PUNor XMT jobname : A job name known to VSE/POWER jobnumber: A job number assigned to job name ALL : Specifies that all jobs in queue are to be released class : Specifies that all jobs in queue for this class are to be released keyword : One of CPRI, CDEST CDISP or CSYSID value : specifies current value of the attribute *abc : Specifies that all jobs in the queue, whose names have the specified characters in common are to be released (generic job name)		PREL000	

NOTES	MODULE	LABEL	REF
1 The 'Verify command authorization' subroutine is called to check if the issuer of the command is authorized. If not, the command is rejected. An appropriate error message has been already issued by the subroutine.			$VCA
2 The 'Verify Queue id' subroutine is called to examine if the operand specifies a valid queue id. Valid queue ids are - RDR, LST, PUN or XMT.			
2A Message 1R52I ccccccccc OPERAND 1 NO VALID QUEUE is issued.		PREL100	$GAM
3 If invalid specifications have been entered message 1R52I ccccccccc OPERAND ## MISSING OR INVALID will be issued.			$GAM
4 The 'VERKEYO' subroutine is called to check if the keyword and the keyword value is valid. If not, the subroutine has already issued an appropriate error message.			
5 state. The disposition is set to 'D' if the original disposition is 'H' or to 'K' if the original disposition is 'L'.			

IPW$$CR - Process Queue update

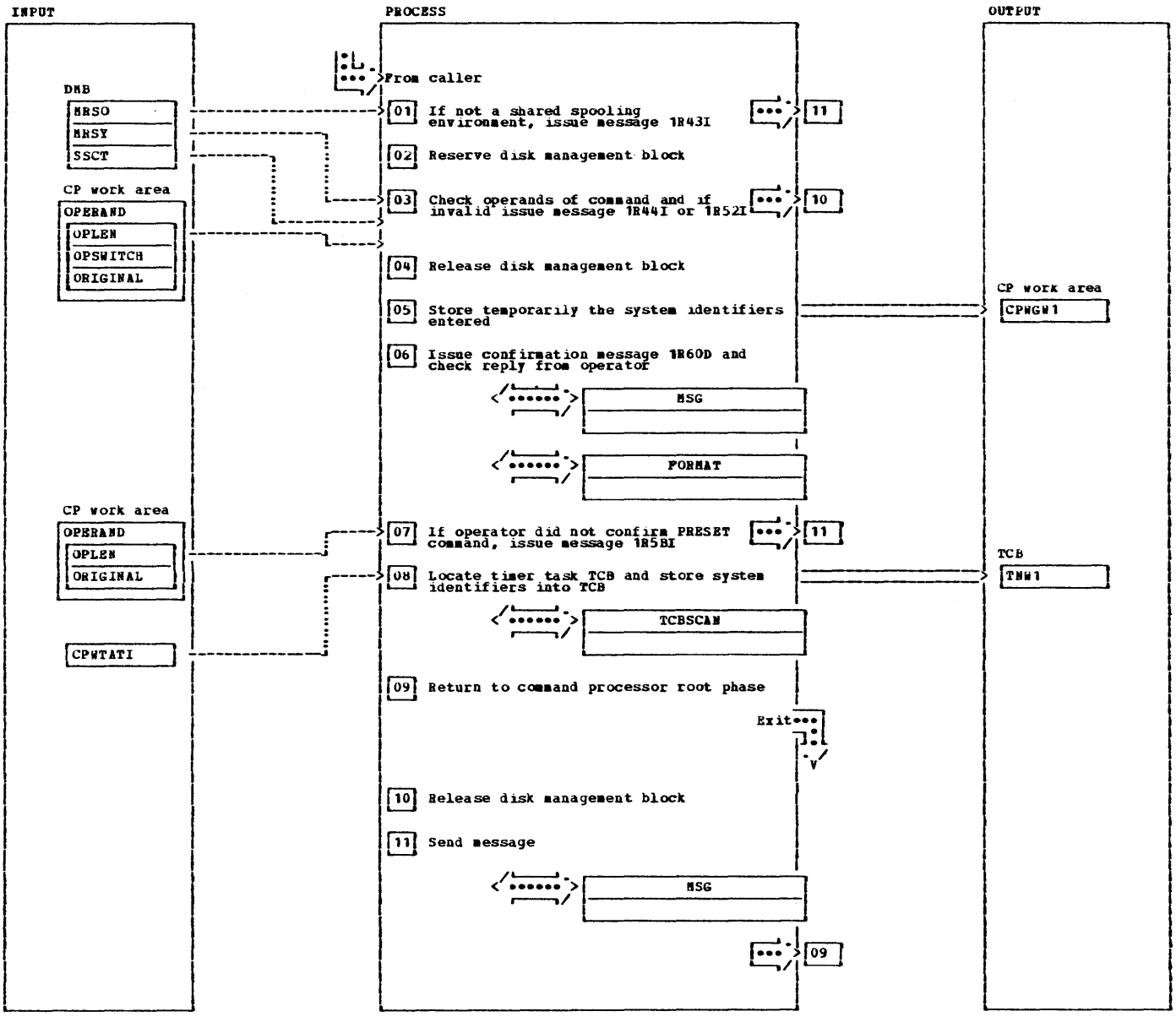


NOTES	MODULE	LABEL	REF
The reader , list , punch and/or xmit queue is scanned and each queue set which is eligible is extracted and updated.			
1 For the queue record area workspace will be provided.		PREL500	$RSW
3B The disk management block is reserved for the duration of the queue scanning.			$RSR
3C Subroutine RELTOABS converts the relative disk address			
3D The first in set queue record is read in the queue record area just acquired.			$RDQ
3E The queue record just read in is examined if it is eligible to be released - jobname and number - class - RJE user id - local or remote destination			

NOTES	MODULE	LABEL	REF
3F When the disposition of the queue set is either 'H' or 'L' it is changed to 'D' or 'K' respectively. Then the first-in-set queue record is written back.			$WTQ
3G Queue sets belonging to a class are chained via the 'QCQN' field. This field contains the absolute disk address of the next queue set in chain. For the last queue set, this field contains binary zeros.			
3H The disk management block is released.			$RLR
5			$RLW
6 If any queue set is released, message 1R88I OK (for the central operator only) or if not 1R88I NOTHING TO RELEASE is issued.			$GAM

This page was left blank intentionally.

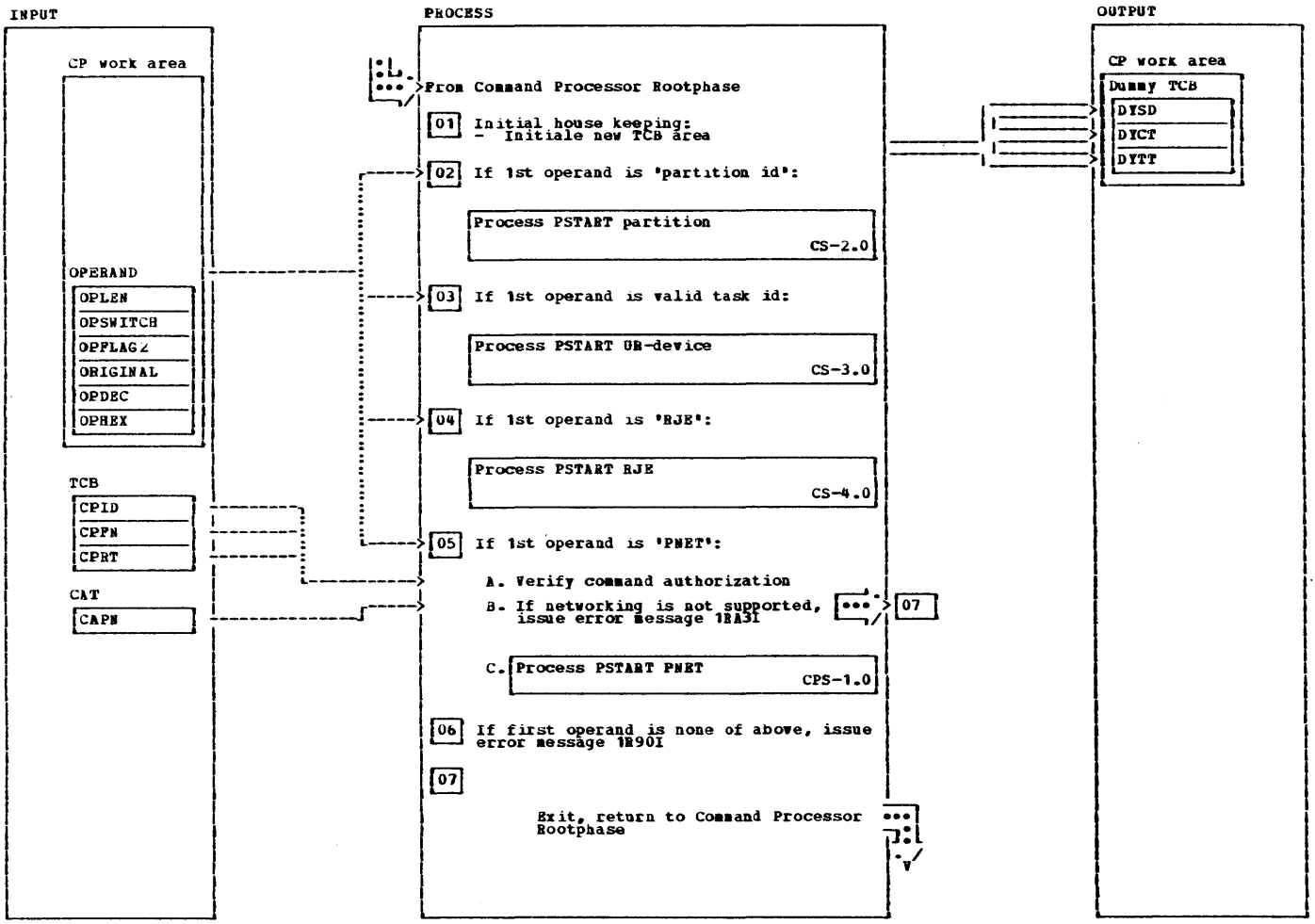
IPW$$CRE - PRESET Processor



IPW$$CRE - PRESET Processor

NOTES	MODULE	LABEL	REF
1 The PRESET command has the following format: PRESET sysid1,sysid2,sysid3,...,sysidn		PCRE000	
1 where: sysid : System identifier of the VSE/POWER system(s) which shall be reset			
1 Note: System identifier must be decimal and 1 byte long. The system identifier must not be the system identifier of the system that issued the PRESET command.			
1 Following message will be issued: 1R43I SHARED SPOOLING NOT ACTIVE			
3 Following messages will be issued: 1R52I command code OPERAND # # MISSING OR INVALID If system identifiers entered are not stored in system table or if the own system identifier was entered, the following message will be issued: 1R44I SYSID n IS OWN OR UNKNOWN		PCRE200	
4			$RLR
6 Following message will be issued: 1R60D CONFIRM PRESET COMMAND FOR SYSID.... Note: The only valid reply from operator is 'YES '		PCRE540	
7 Following message will be issued: 1R5BI COMMAND IGNORED		PCRE580	
8 Task identifier for timer task is 'T TI'		PCRE100	$RSR
1			$RLR
11			$GAR

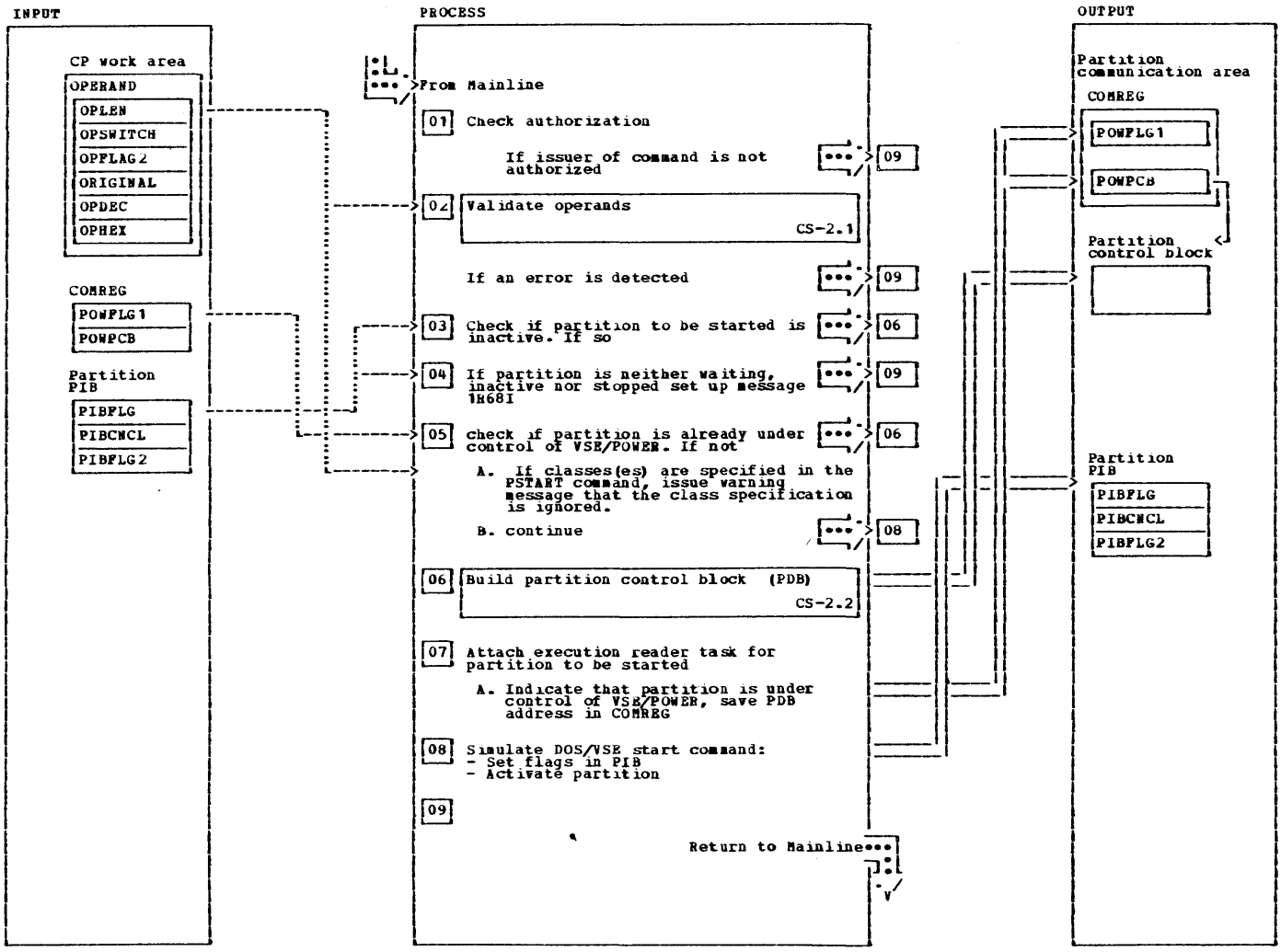
IPW$$CS - PSTART Processor



NOTES	MODULE	LABEL	REF
1 The new TCB is initialized in the dummy TCB area contained in the CP-workarea. Following fields are setup: - storage descriptor - class list - termination code.		PSTRT00	
2 The 'VPARTID' subroutine is called to check whether the first operand is a valid partition id or not. On return of the subroutine, field 'OTHPIBPT' contains the address of the PIB of the partition supposed to be started or is hex zero, if the first operand is not a valid partition id.			
3 The 'VTASKID' subroutine is called to see whether the first operand is a valid task id or not. Valid task id's are 'RDR', 'LST' , and 'PUN'.			

NOTES	MODULE	LABEL	REF
5A The authority associated with the issuer of the command is examined. If the authority is insufficient, the subroutine issues an appropriate error message.		PSTPNET0	$VCA
5B Message 1RA3I cccccccc VSE/POWER NETWORKING NOT SUPPORTED is issued.			$GAM
6 If invalid task specification was entered, the following message will be issued: 1R90I cccccccc INVALID TASK SPECIFICATION xxxxxxxx			$GAM

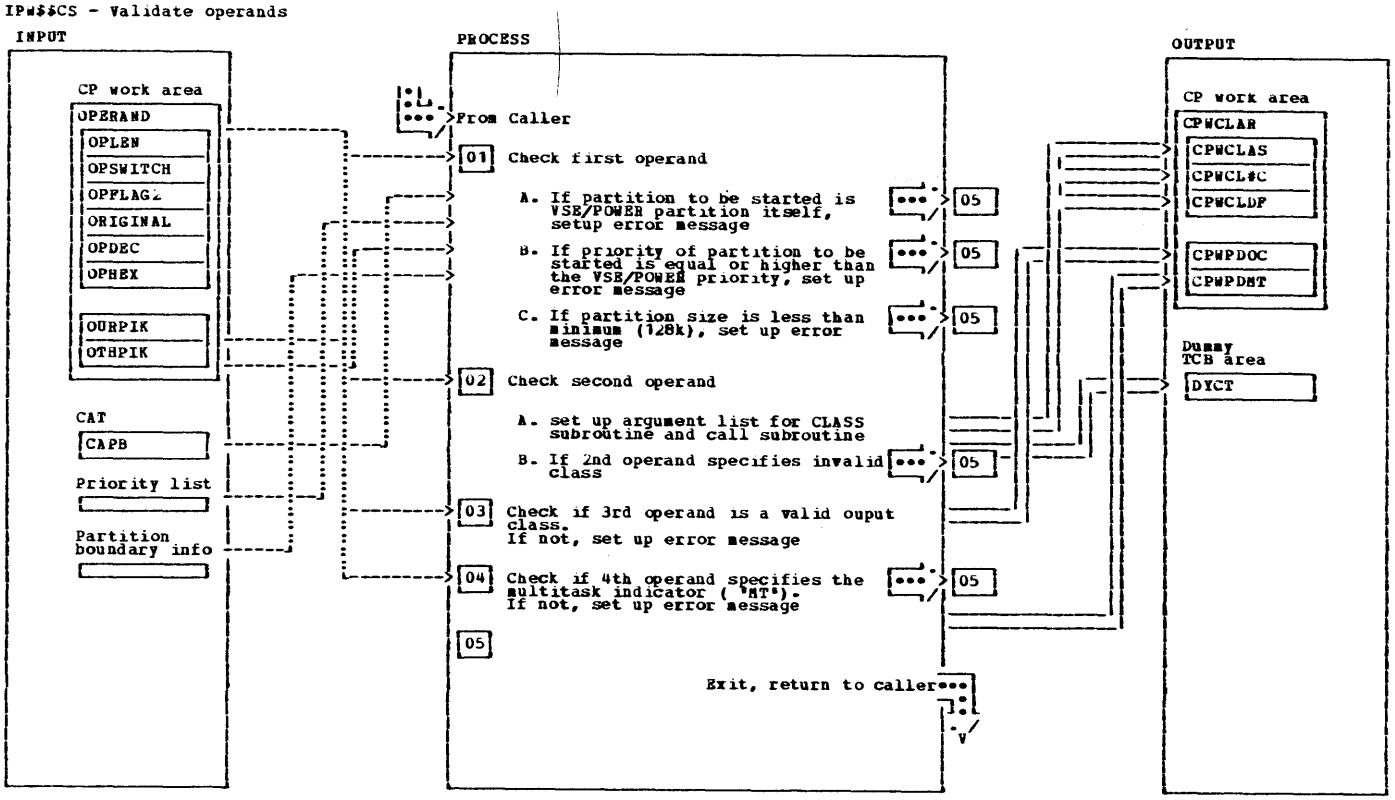
IPW$$CS - PSTART partition Processor



NOTES	MODULE	LABEL	REF
1 The authority associated with the issuer of the command is examined. This is done by invoking the appropriate subroutine contained in the Rootphase. If the authority is insufficient, the subroutine issues an appropriate error message.		PSTP100	$VCA
3	GETFLD	PSTP500	
4 The partition supposed to be started must be in one of the following conditions. If not, the partition is not available. - inactive - waiting and in stopped state. Message 1R68I xx PARTITION NOT AVAILABLE is issued.			$GAM
5 If the stopped partition was already under control of VSE/POWER an execution reader task exists already for the partition. Therefore it is not neccessary to create a partition control block and to attach an execution reader task again			

NOTES	MODULE	LABEL	REF
5A Message 1R80I WARNING: CLASS SPECIFICATION IGNORED is issued			$GAM
7 The 'ATTACH' subroutine, contained in the command processor rootphase, is called to reserve storage for the new execution reader task TCB and finally to attach the task.			
8 For an inactive partition, the pointer to the partition save area , which is identical to the partition start address is saved in the partition PIB. The partition save area is zeroized and the default name 'NO NAME' is put into the save area as program name and in the COMREG as job name			
For a partition which is currently in stopped state, the stopped state flag is reset, the JC-flag is set in the PIB which causes to invoke end-of-job processing and the partition is made dispatchable	THEADY		
During updating PIB interrupts are disabled, the DOS/VSE system is shortly seized and released again	SVC22	PSTP540	

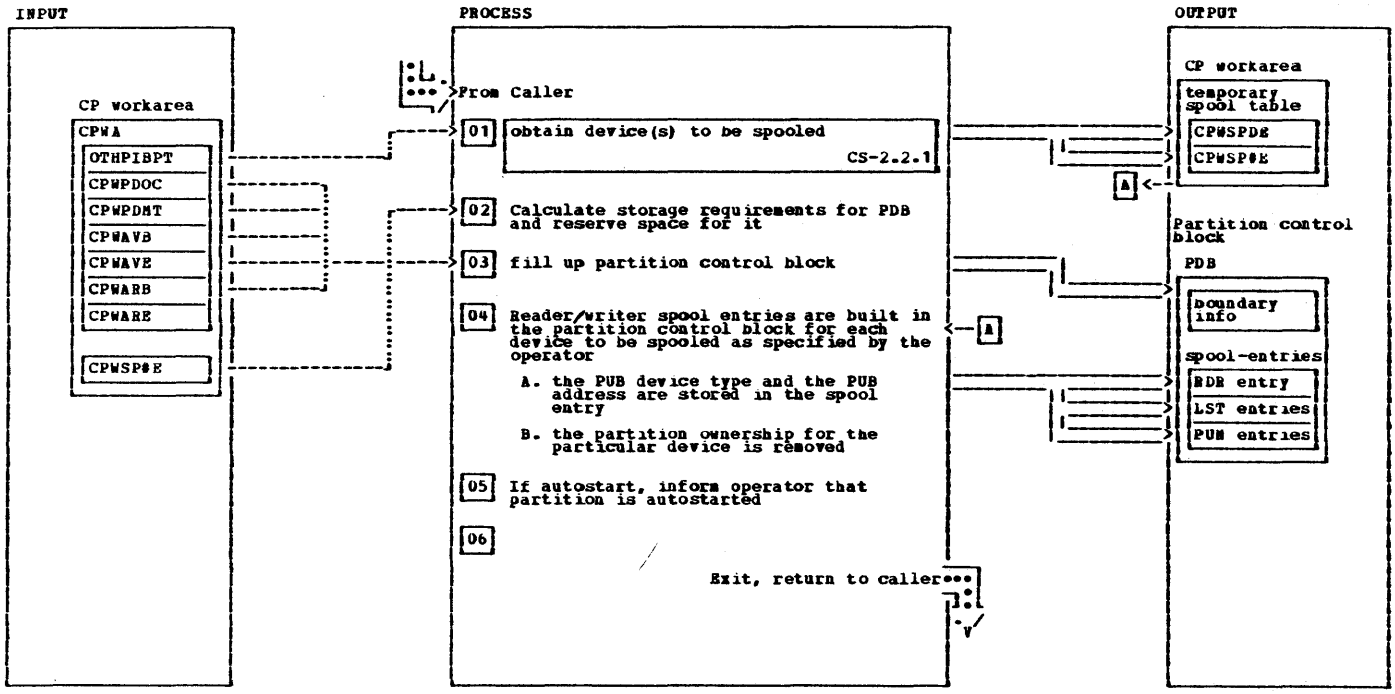
IPw$$CS - Validate operands



	NOTES	MODULE	LABEL	REF
1	The first operand specifies the partition to be brought under control of VSE/POWER			
1A	1R68I cccccccc xx IS THE VSE/POWER PARTITION			
1B	The GETPRTY macro is issued to get the priority list The partition interrupt key (PIK) must occur in the priority list left from the VSE/POWER PIK nor must the found PIK be separated by the '=' sign from the VSE/POWER PIK. If not, message 1R63I cccccccc xx PRIORITY TOO HIGH is issued	GETPRTY	PSTP120	$GAM
1C	The EXTRACT macro is issued to get the partition boundary information. The partition size is calculated and checked if less than minimum (currently 128K). If so, message 1R72I command code VIRTUAL xx IS SMALLER THAN 128K is issued	EXTRACT		$GAM
2	The 2nd operand specifies, if not omitted, classes of the RDR queue to be processed by the partition to be brought under control of VSE/POWER. If the 2nd operand is omitted, class 'A' is assumed as default		PSTP200	
2A	The parameter list for the CLASS subroutine is set up: - task type (EX reader) - max. # of classes allowed - 2nd operand (classes)			

	NOTES	MODULE	LABEL	REF
	The CLASS subroutine checks if valid class(es) are specified in the command. Valid calsses are 'A' - 'Z' as well as the partition dependent class (eg. 0 for BG). The return code , given back by the CLASS subroutine in register 15, tells the issuer if an invalid class was specified. In this case an error message has been already issued the CLASS subroutine stores the class table pointers in the TCB class list			
3	The 3rd operand specifies, if not omitted, the default output class assigned to all output queues sets. If the operand is omitted, class 'A' is taken as default. Only one class specification is allowed, valid specifiactions are 'A' - 'Z'. Message 1R52I OPERAND ## INVALID OR MISSING is issued		PSTP300	$GAM
4	The 4th operand, if not omitted. specifies the multitask indicator, which must be 'MT'. If not 'MT', message 1R52I OPERAND ## INVALID OR MISSING is issued		PSTP400	$GAM

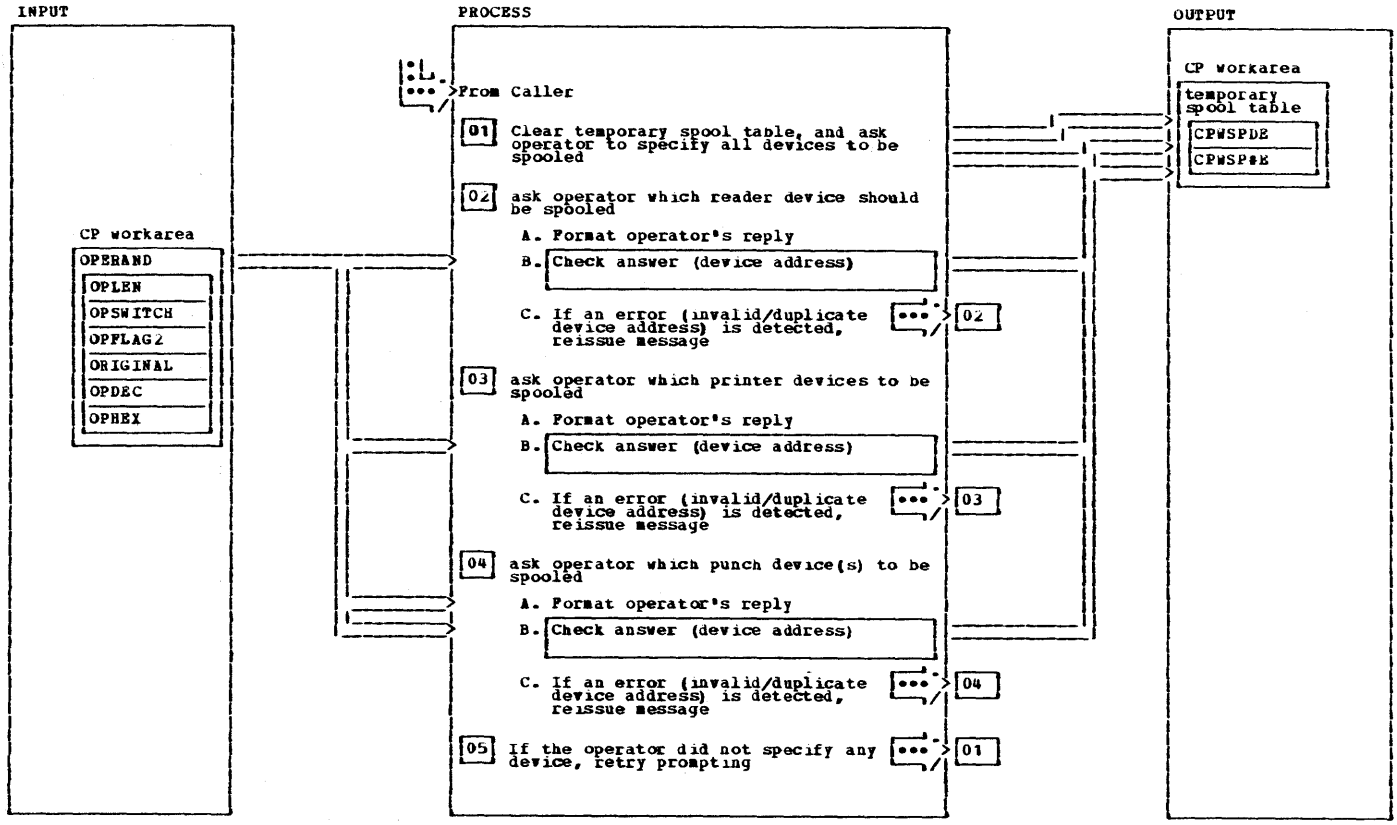
IPW$$CS - Build partition control block (PDB)



NOTES	MODULE	LABEL	REF
This routine prompts the operator to specify all UR-devices to be spooled for the partition to be brought under control of VSE/POWER. It calculates the required storage amount and creates within the PDB for each device a spool entry.		PSTBP00	
1 the operator is prompted to specify the UR devices to be spooled for the partition to be started			

NOTES	MODULE	LABEL	REF
3 The partition control block is built and filled up with following information: - PIB address - output class - multitask indicator (if applicable) - partition boundary information			$RSW
5 message 1R75I xx AUTOSTARTED is issued			$GAM

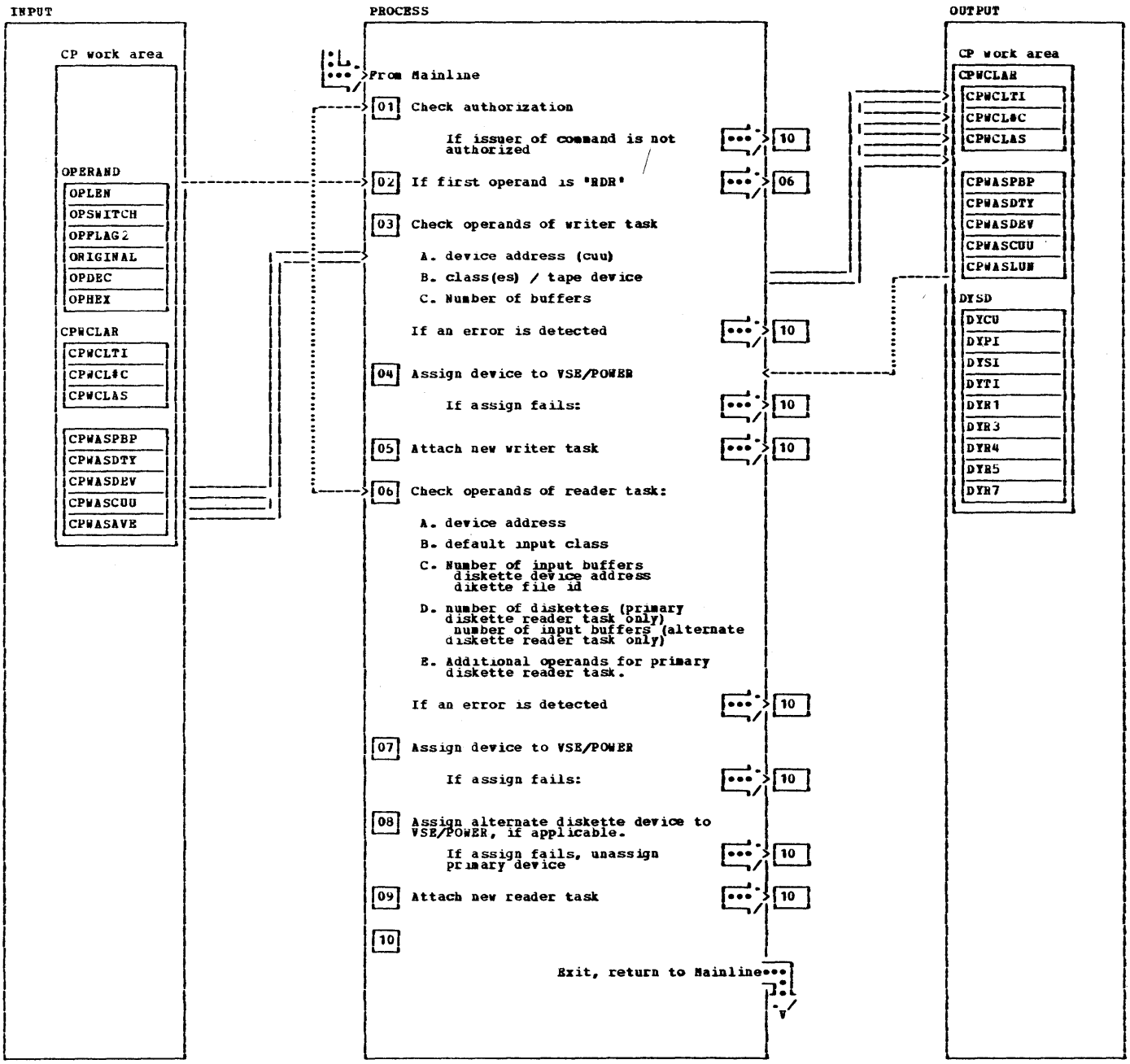
IPW$$CS - Get device(s) to be spooled from operator



NOTES	MODULE	LABEL	REF
1 message 1R861 PLEASE SPECIFY DEVICES TO BE SPOOLED is issued			$GAM
message 1R861 xx READER= is issued the reply of the operator is formatted and validated. When an error is detected, the message is re-issued			$GAM
message 1R861 xx PRINTER= is issued			$GAM

NOTES	MODULE	LABEL	REF
the reply of the operator is formatted and validated. When an error is detected, the message is re-issued			
Message 1R861 xx PUNCHES= is issued The reply of the operator is formatted and validated. When an error is detected, the message is re-issued			$GAM
5 when the operator answered three times 'NO', he did not specify any device			

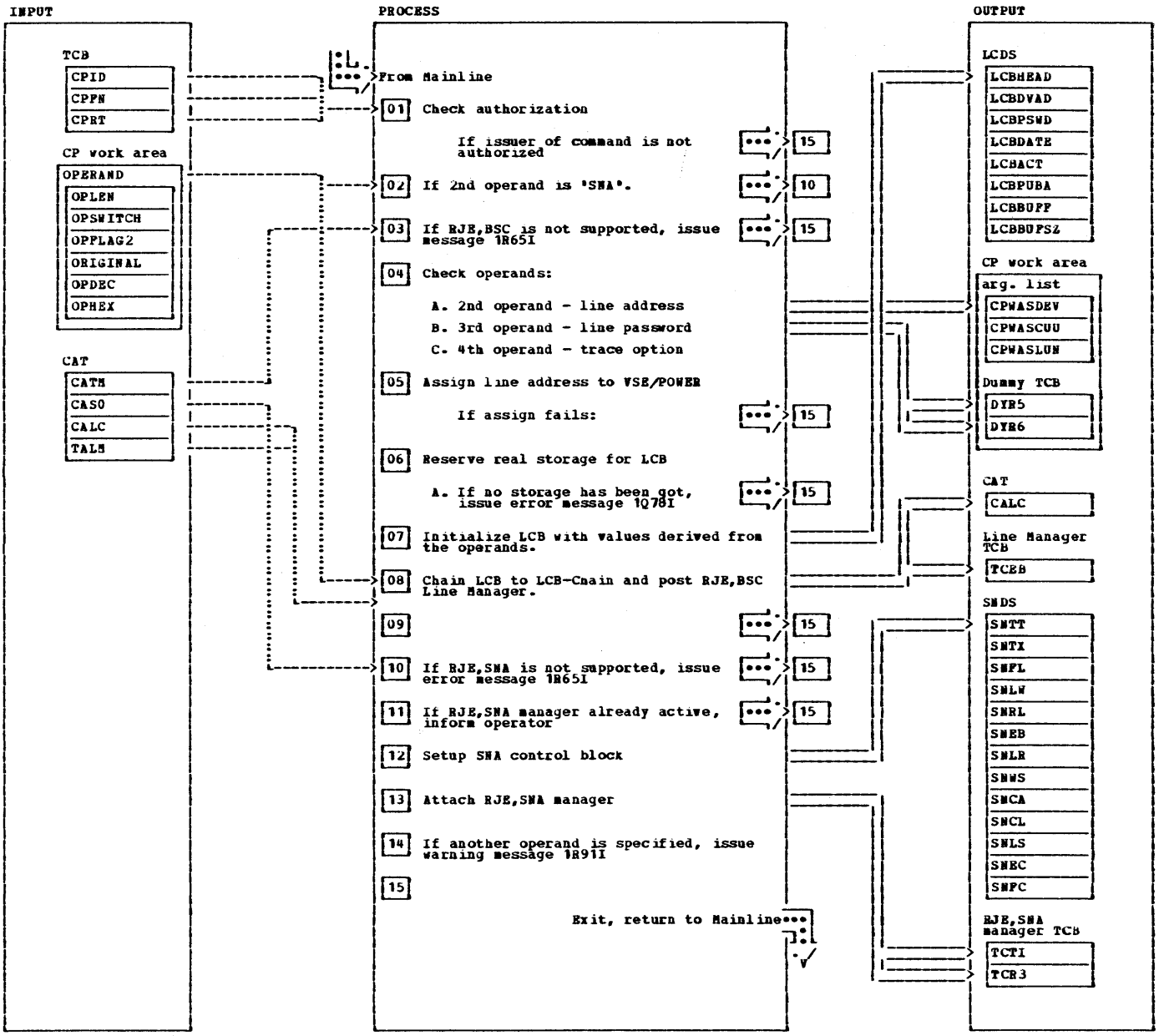
IPW$$CS - PSTART,task Processor



NOTES	MODULE	LABEL	REF
1 The authority associated with the issuer of the command is examined. This is done by invoking the appropriate subroutine contained in the Rootphase. If the authority is insufficient, the subroutine issues an appropriate error message.		PSRW100	$VCA
3 The 2nd operand is mandatory and specifies the printer/punch device address. Valid specification are cuu or X'cuu'. The 3rd operand, if not omitted, specifies either the classes to be processed by the writer task or the tape device address in case of a tape writer task. Valid class specification are 'A' - 'Z'. Up to four classes are allowed. The 'CLASS' subroutine, contained in the command processor rootphase is called to check if the class specification is valid or not. In case of invalid class specification, subroutine CLASS has already issued error messages, so a branch is taken to processor exit. To distinguish between classes and tape address, the tape address must be specified in hex notation. The 4th operand, if not omitted, specifies the number of output buffers to be used. Valid specification are '1', '2' or 'D'. The operand is not applicable for a punch task. If an error is detected one of the following mesages is issued: 1R74I ccccccccc INVALID DEVICE SPECIFICATION 1R52I ccccccccc INVALID BUFFER SPECIFICATION 1R58I ccccccccc DEVICE cuu IS NOT KNOWN 1R52I ccccccccc OPERAND ## MISSING OR INVALID 1R42I ccccccccc OPERAND ## INCORRECT			$GAM
4 The 'ASSIGN' subroutine is called to check whether the specified device is known, available, and valid for the specified task type. If the assign fails, the subroutine has already issued an appropriate error message.		PSWTM800	
5 The 'ATTACH' subroutine is called to reserve storage for the new writer task TCB, to initialize the TCB with values stored tempory in the dummy TCB area and finally to attach the new task.			

NOTES	MODULE	LABEL	REF
6 The 2nd operand is mandatory and specifies the reader device address, which can be either a card reader, a tape or a diskette device. The specification must be in 'cuu' or X'cuu' format. The PUB is scanned to obtain the device type. The 3rd operand specifies the input class. Valid class specification are '0' - '9', 'A' - 'Z'. If the operand is omitted, class 'A' is taken. The 'CLASS' subroutine is called to validate the class specification. The 4th operand has different meaning depending on the type of reader task. The operand is not applicable for a tape reader task. If this is a card reader task, the operand can either be the number of input buffers or the alternate diskette device address which must be defined in hex notation. Valid buffer number specification is '1' or '2'. If an error is detected one of the following mesages is issued: 1R74I ccccccccc INVALID DEVICE SPECIFICATION 1R52I ccccccccc INVALID BUFFER SPECIFICATION 1R58I ccccccccc DEVICE cuu IS NOT KNOWN 1R52I ccccccccc OPERAND ## MISSING OR INVALID 1R42I ccccccccc OPERAND ## INCORRECT		PSRDR200 PSMDR200	 $GAM
7 The 'ASSIGN' subroutine is called to check whether the specified device is known, available, and valid for the specified task type. If the assign fails, the subroutine has already issued an appropriate error message.			
9 The 'ATTACH' subroutine is called to reserve storage for the new reader task TCB, to initialize the TCB with values stored tempory in the dummy TCB area and finally to attach the new task.			

IPW$$CS - PSTART,RJE Processor

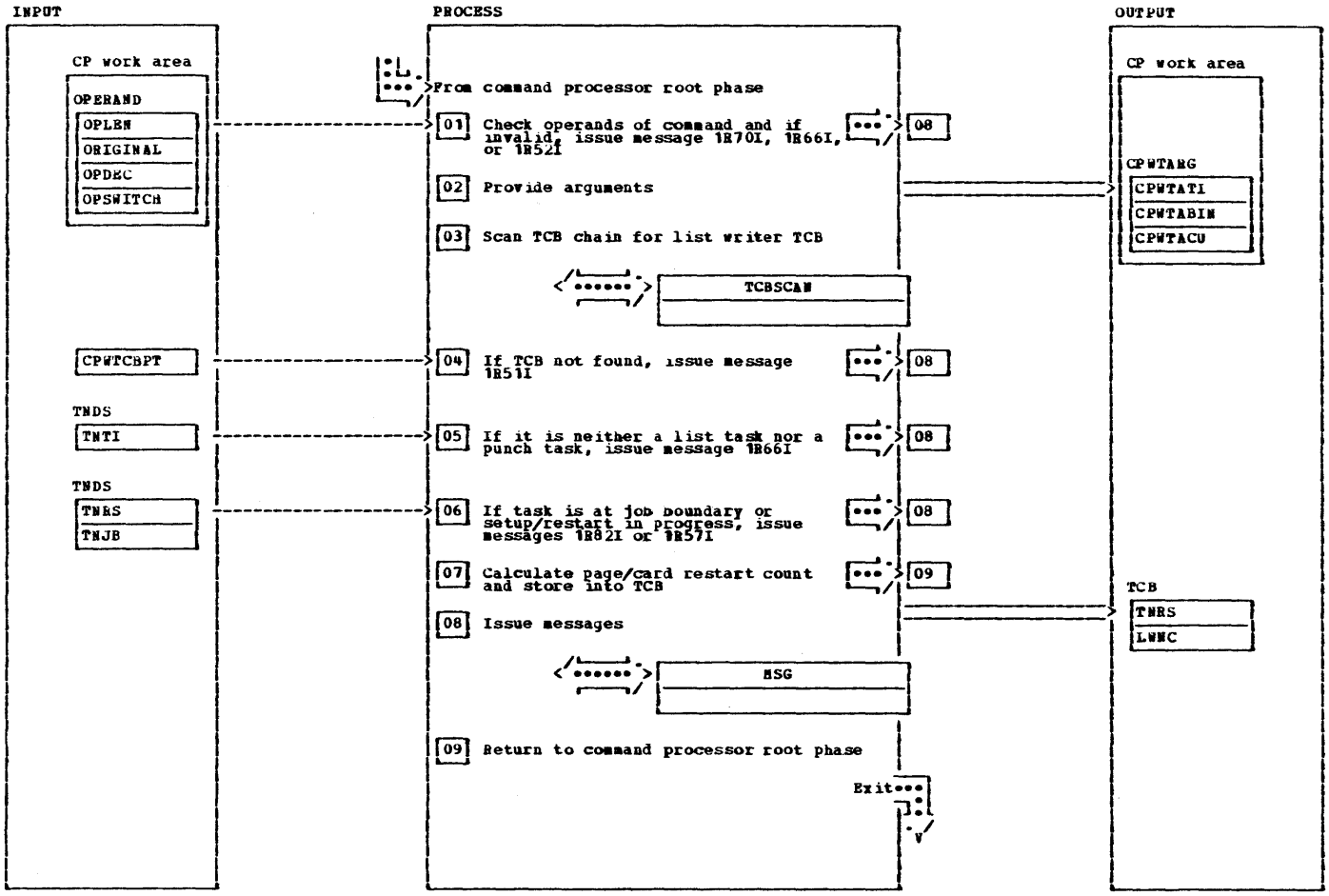


IPW$$CS - PSTART,RJE Processor

NOTES	MODULE	LABEL	REF
1 The authority associated with the issuer of the command is examined. This is done by invoking the appropriate subroutine contained in the Rootphase. If the authority is insufficient, the subroutine issues an appropriate error message.		PSTRJE10	$VCA
3 Message 1R65I ccccccccc RJE,BSC NOT SUPPORTED is issued.			$GAM
4 One of the following messages will be issued, if an error is detected: 1R52I ccccccccc OPERAND ## INVALID 1R74I ccccccccc INVALID LINE ADDRESS 1R62I ccccccccc INVALID RJE PASSWORD			$GAM
5 The 'ASSIGN' subroutine is called to check whether the specified line address is a valid RJE device or not and to assign the RJE line to VSE/POWER. If the assign fails, the subroutine has already issued an appropriate error message.		PSTRJE30	
6 Real storage is reserved for the Line Control Block (LCB).			$RSW
6A Message 1078I NO REAL/PFIXED STORAGE AVAILABLE FOR ttttt,cuu			$GAM
7 LCB identifier is 'LCB'. The following fields are initialized: LCBHEAD = Line address cuu LCBDVAD = Line address cuu LCBCID = Account identifier LCBDATE = System Date YY/MM/DD LCBPSWD = Line password LCBPUBA = PUB address of line LCBLUBN = Logical Unit Number. The real address of the read ans write buffer is calcaluted and stored into the LCB. The LCBACT byte is flagged to indicate the Line Manager start of the line. If 'TRACE' was specified, LCBACT is flagged to indicate the Line Manager tracing required.			

NOTES	MODULE	LABEL	REF
8 The LCB is chained as first in chain to the LCB-Chain via Compare and Swap (CS) Instruction. The address of the Line Manager TCB is loaded and the ECB is posted. If the LM is inactive, not any start was performed previously, the task is dispatched to perform the required activities.			
10 Message 1R65I ccccccccc RJE,SNA NOT SUPPORTED is issued.		PSTSNA10	$GAM
11 Message 1R65I ccccccccc RJE,SNA ALREADY STARTED is issued.			$GAM
13 The 'ATTACH' subroutine is called to attach the RJE,SNA manager. The task identifier is 'LSNA'.			
14 Message 1R91I ccccccccc TOO MANY OPERANDS, FIRST n PROCESSED. is issued			$GAM

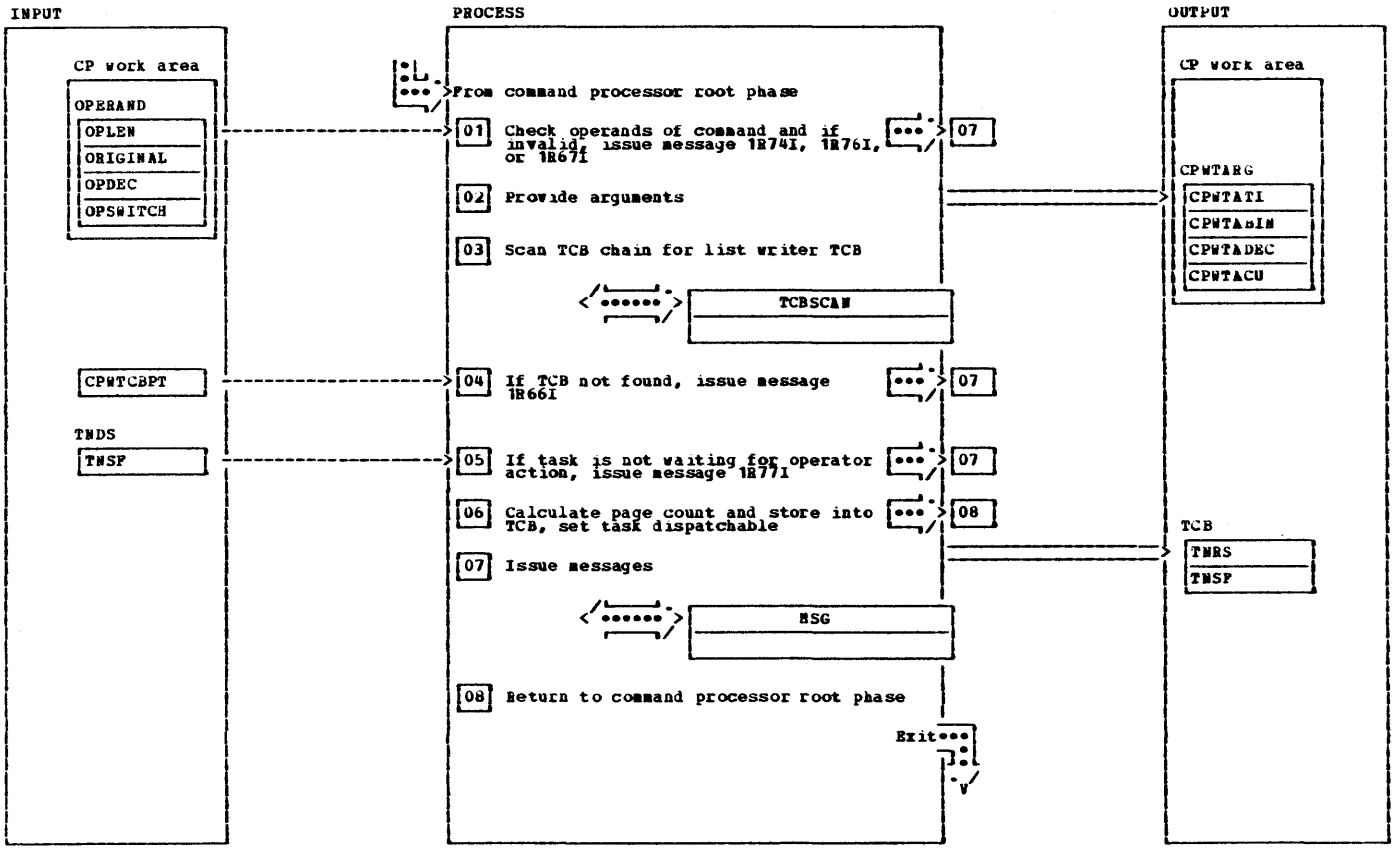
IPW$$CT - PRESTART Processor



NOTES	MODULE	LABEL	REF
1 The PRESTART command has following format: PRESTART uraddr<,n><,i> 1 where: uraddr : Specifies a unit record address in the form cuu or x'cuu' n : Specifies the number of cards/pages i : Specifies the copy group index 1 If invalid specification has been entered, one of the following messages will be issued: 1R70I command code NO DEVICE ADDRESS SPECIFIED 1R66I command code NO WRITER TASK SPECIFIED 1R52I command code OPERAND 2 NEITHER DECIMAL NOR OMITTED 1R52I command code OPERAND 3 INVALID		PRST100	

NOTES	MODULE	LABEL	REF
3 Task identifier for list writer TCB is 'WLST'. 4 Following message will be issued: 1R51I command code OPERAND 1 DESIGNATES NON-EXISTING TASK 5 Following message will be issued: 1R66I command code NO WRITER TASK SPECIFIED 6 One of the following messages will be issued: 1R82I command code PSETUP OR PRESTART IN PROGRESS 1R57I command code COMMAND IGNORED, TASK AT JOB BOUNDARY 8			$SAM

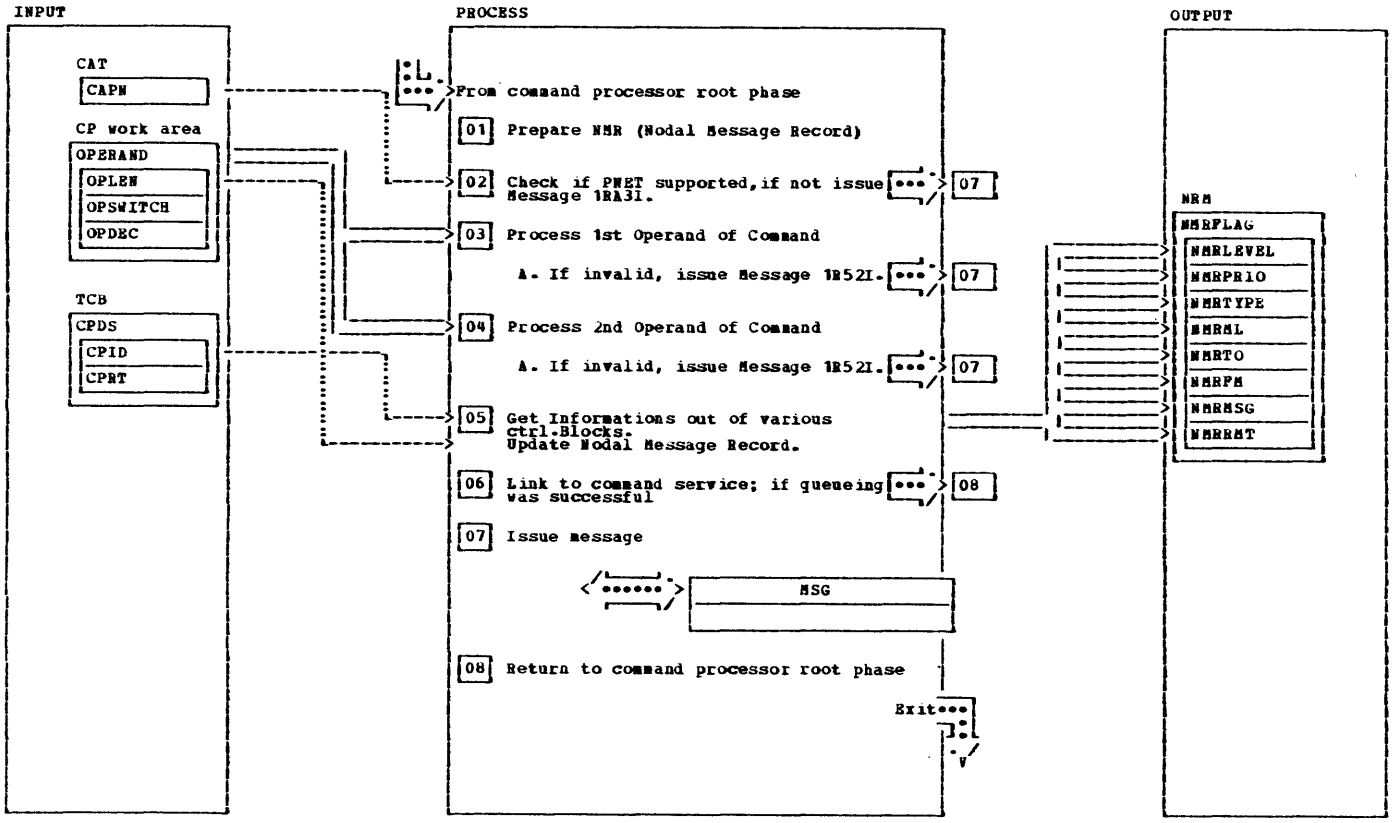
IPW$$CU - PSETUP Processor



NOTES	MODULE	LABEL	REF
1 The PSETUP command has following format: PSETUP uraddr<,n>		PSETP00	
1 where: uraddr : Specifies a unit record address in the form cuu or x'cuu' n : Specifies the number of pages			
1 If invalid specification has been entered, one of the following messages is issued: 1R74I command code NO PRINTER ADDRESS SPECIFIED 1R75I command code NUMBER OF PAGES NOT DECIMAL 1R67I command code OPERAND REDUCED TO 99 NOTE: Message 1R67I is a warning message only.			

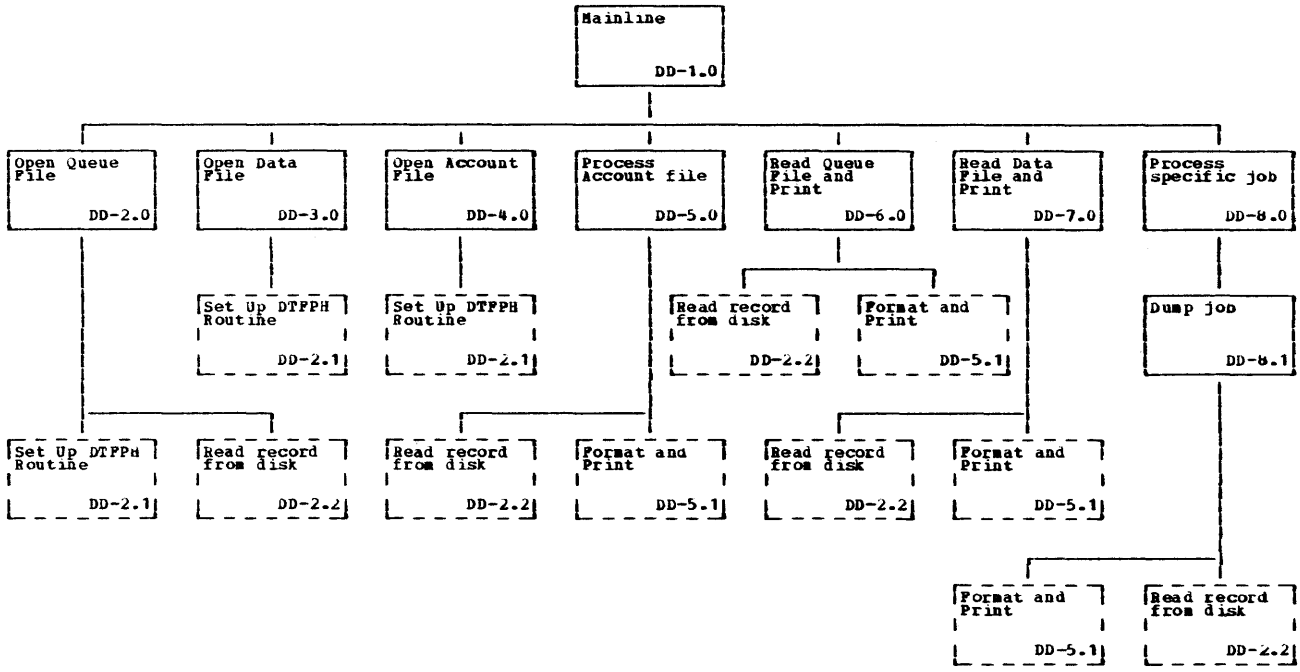
NOTES	MODULE	LABEL	REF
3 Task identifier for list writer TCB is 'WLST'.			
4 Following message will be issued: 1R66I command code LIST WRITER TASK DOES NOT EXIST			
5 Following message will be issued: 1R77I command code TASK NOT WAITING FOR OPERATOR			
7			$GAM

IPW$$CX - PXMIT Processor

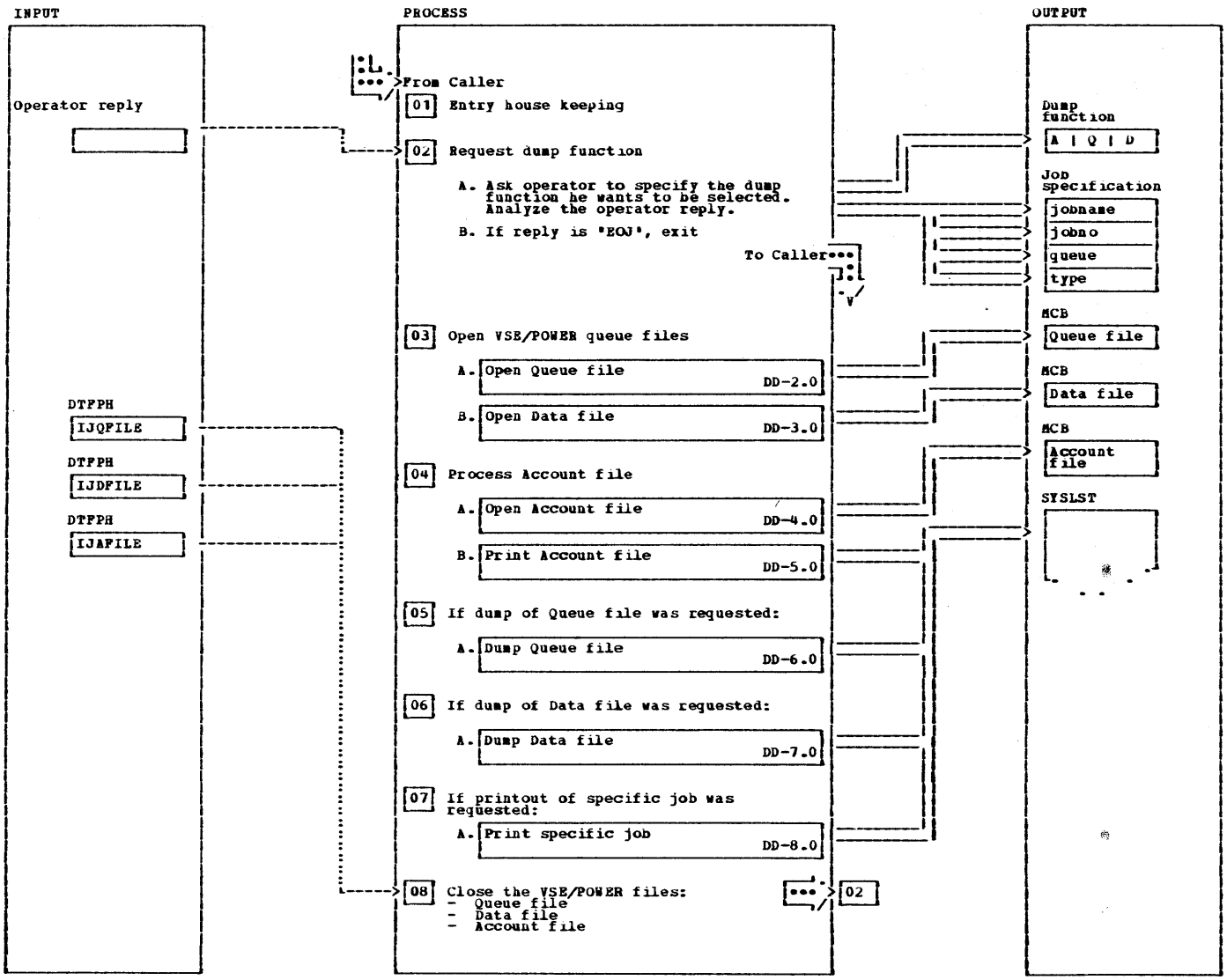


NOTES	MODULE	LABEL	REF
2 The following message will be issued: 1RA3I VSE/POWER NETWORKING NOT SUPPORTED. 3 The PXMIT command has the following format: PXMIT nnnnnnnn,cmnd where is: nnnnnn : A valid node identifier of the system to which the command is to be sent. cmnd : Any command which is valid on the target system. The following message will be issued: 1R52I command code OPERAND 1 MISSING OR INVALID Note: It is not checked here wether the node is known to the system nor if there is a connection to this node.This is be done later by the service subroutine.			

NOTES	MODULE	LABEL	REF
4 The following message will be issued: 1R52I command code OPERAND 2 MISSING OR INVALID Note: The validity of the command to be transmitted to the node is not checked here. This is done later by the target system. 6 Following returns are possible: 0(R1) - normal return 4(R1) - no storage 8(R1) - node unknown C(R1) - node unreachable 7 Following messages, depending on the RC are possible: 1Q7AI ccccc NO VIRTUAL STORAGE AVAILABLE 1RB1I NO CONNECTION ESTABLISHED TO NODE nnnnnnnn			$ICS $GAM



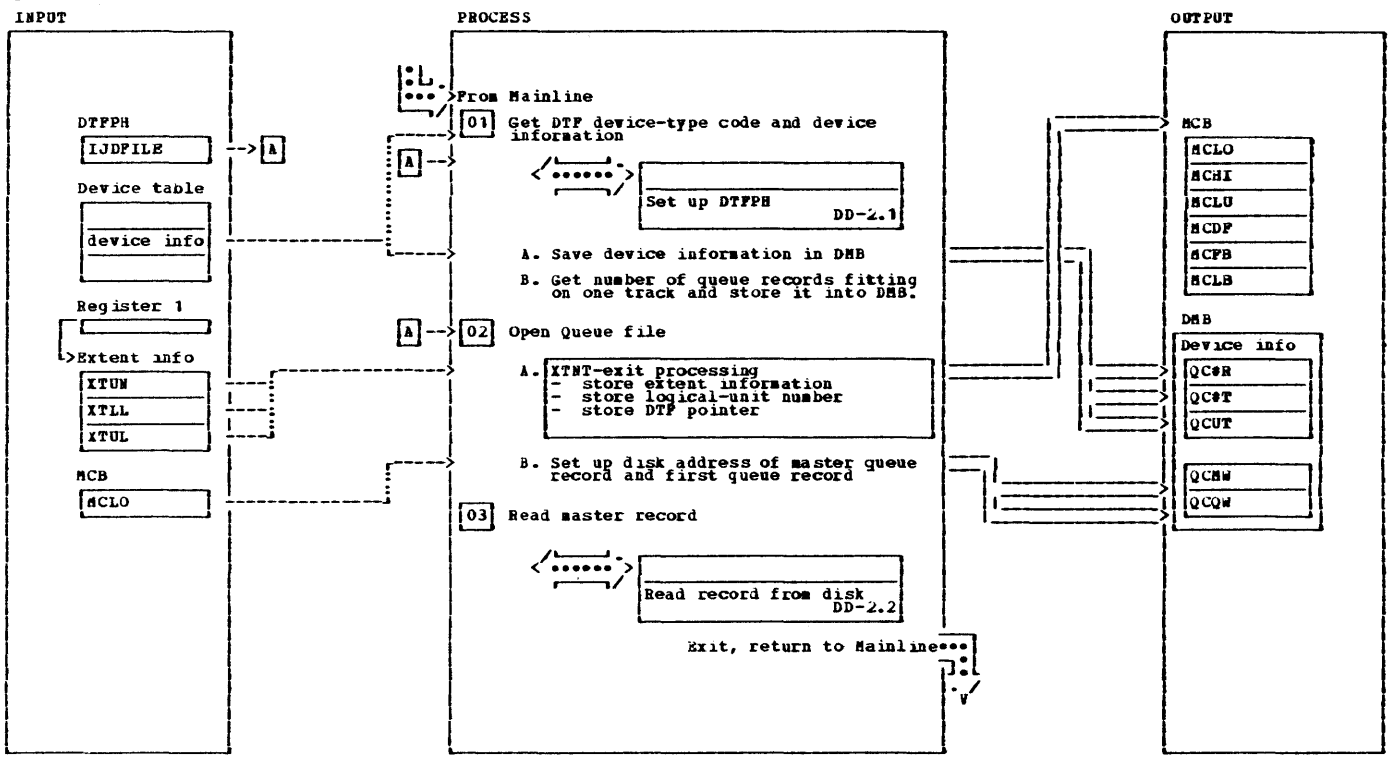
IPW$$DD - Mainline



IPW$$DD - Mainline

	NOTES	MODULE	LABEL	REF
	The program IPW$$DD enables the user to display the queue file, the data file or the account file. The user may select a complete dump of one of the files or, for the queue file and data file, only the entries that belong to a specific job entry.			
1	The base registers are set up and the default SYSLST line size is obtained from the communication area.	COMRG		
2	The operator is prompted to specify the dump function he wants to be done. Message DUMP FUNCTION= is issued.	EXCP	DD10	
	If nothing has been entered, message INVALID REPLY is issued and the operator is re-prompted.	WAIT		
	If the reply is not 'Q' nor 'D' nor 'A', it is assumed to be a jobname.			
2A	The jobname and if specified the jobnumber are validated: - jobname must be alphameric, 1 - 8 characters - jobnumber numeric, maximum of six digits.			
	If an invalid jobname has been specified, message INVALID JOBNAME is issued and the operator is re-prompted. Otherwise the jobname and its length is saved. If an invalid jobnumber has been specified, message INVALID JOBNUMBER is issued and the operator is re-prompted. Otherwise the jobnumber is saved.			
	If a third operand is specified, it indicates the queue to be looked at. Valid specifications are 'R', 'L', 'P' or 'X'. If none of them has been entered, message INVALID THIRD OPERAND is issued and the operator is re-prompted.			
	If a fourth operand is specified, which is only valid when the third operand was 'X', it indicates in which sub-queue the queue set is supposed to be. Valid specifications are 'JOB' or 'OUT'. If none of them has been entered or the 4th operand is not applicable, message INVALID FOURTH OPERAND is issued and the operator is re-prompted.			
2B	If the operator replied 'EOJ', the DOS/VSE EOJ macro is issued to terminate the process.	EOJ		
8	If the Queue file is open, it is closed.	CLOSER		
	If the Data file is open, it is closed.	CLOSER		
	If the Account file is open, it is closed.	CLOSER		

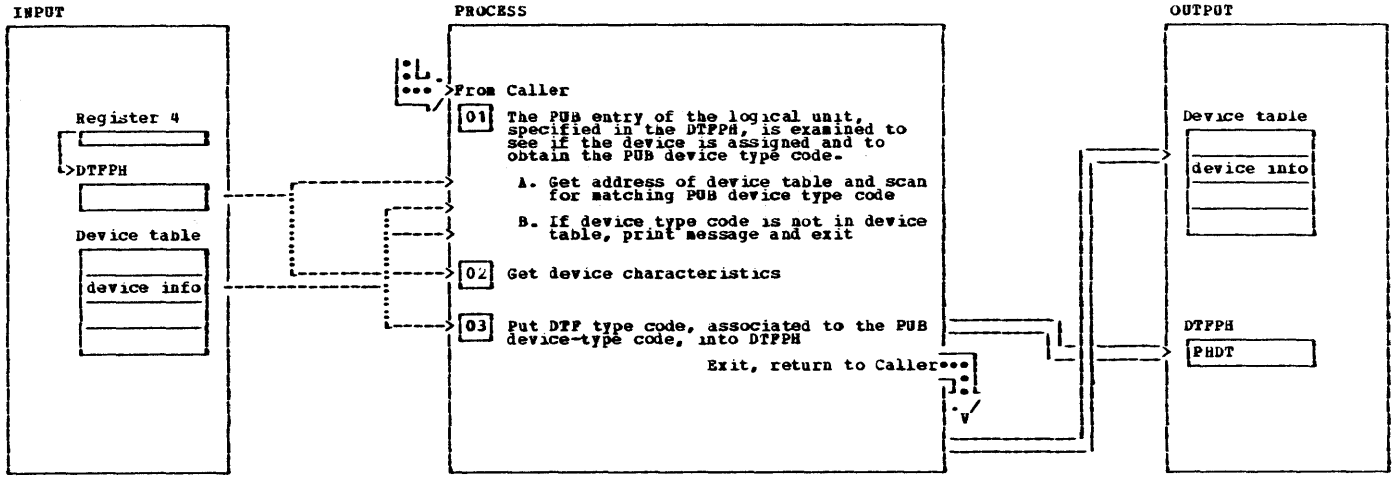
Open Queue file



NOTES	MODULE	LABEL	REF
The Queue file is opened for input operations. All relevant fields in the DMB and MCB are initialized.			
1 The 'Set up DTFPH' routine returns the address of the device-table entry associated with the device.		DD52	
1A All device-dependent information is saved in the DMB: - number of tracks/cylinder - number of blocks (FBA only)			
1B	GETVCE		
2 The Queue file is opened by issuing the OPENR macro instruction.	OPENR		
2A Control is given to this routine when OPEN is performed by IOCS. The extent information (lower & upper limit) is saved in the MCB.		DD70	
2A The logical-unit number and the DTF pointer are saved in the MCB.			

NOTES	MODULE	LABEL	REF
2A If the Queue-file resides on an FBA device, the relative ending block number is calculated for the extent. This is done by subtracting the physical starting block number from the physical ending-block number. (Note: the relative block number is always zero). Control is returned to the calling routine via the LBRET 1 macro intruction.	LBRET		
2B The disk address is set up to address the first queue record which is in fact record number 2 (the first queue record is used for internal purposes only). Seek address is lower extent limit and record number is 2 for CKD; for FBA, the block number is set to one.			
2B The master record lies on the end of the queue file and occupies four normal queue records.			

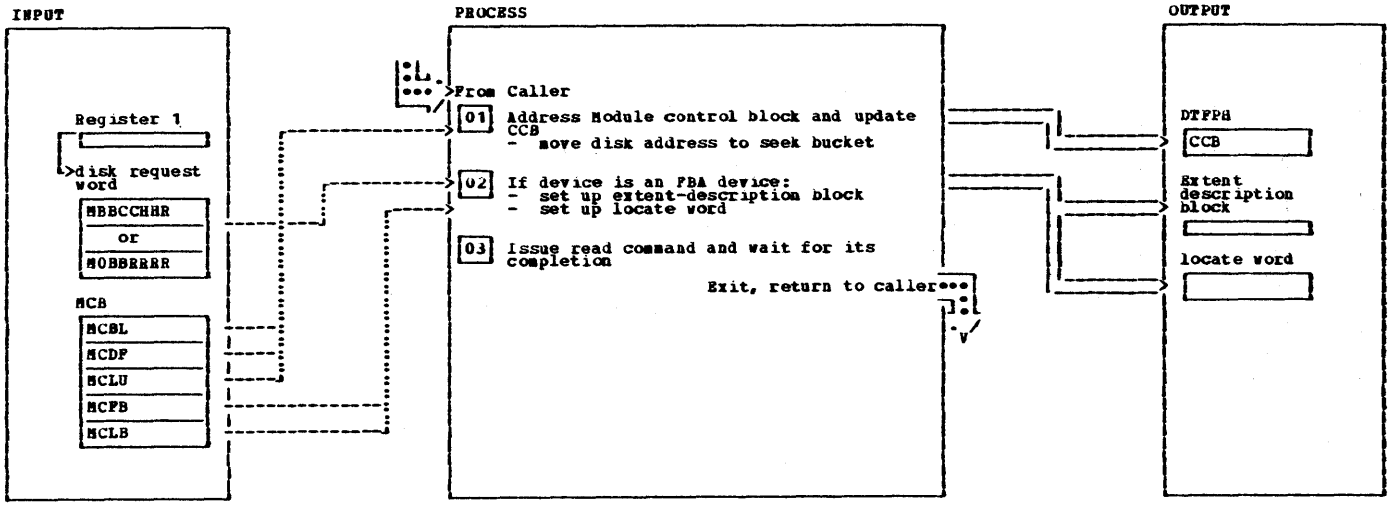
Set up DTFPH (Subroutine)



NOTES	MODULE	LABEL	REF
The PUB device-type code is obtained from the logical unit specification in the DTFPH. This code is checked against the supported device-type codes in the VSE/POWER device table. If no match is found, message INVALID LOGICAL UNIT is issued and a branch to taken to exit the process. If a match is found, the proper device-type code is inserted in the DTFPH.		DD62	
Then the SYSIR macro is issued in order to get the applicable PUB and LUB address for the logical unit.	SYSIR		
1 The EXTRACT macro is issued in order to examine if the logical unit is assigned and the obtain the PUB device type code. If a non-zero return code is given back, it is assumed that the logical unit is not assigned. I this case error message INVALID LOGICAL UNIT is issued and a branch is taken to close the files.	EXTRACT		

NOTES	MODULE	LABEL	REF
1A The internal device table, containing one entry for each disk device supported by VSE/POWER, is scanned to locate the entry with the matching PUB device-type code.			
1B If the device-type code is not in the device table, message INVALID LOGICAL UNIT is issued and a branch is taken to close the files.			
2 A GETVCE macro is issued in order to obtain the disk device information (eg. no of tracks/cyl.)	GETVCE		
3 Insert DTFPH device-type code into DTFPH		DD64	

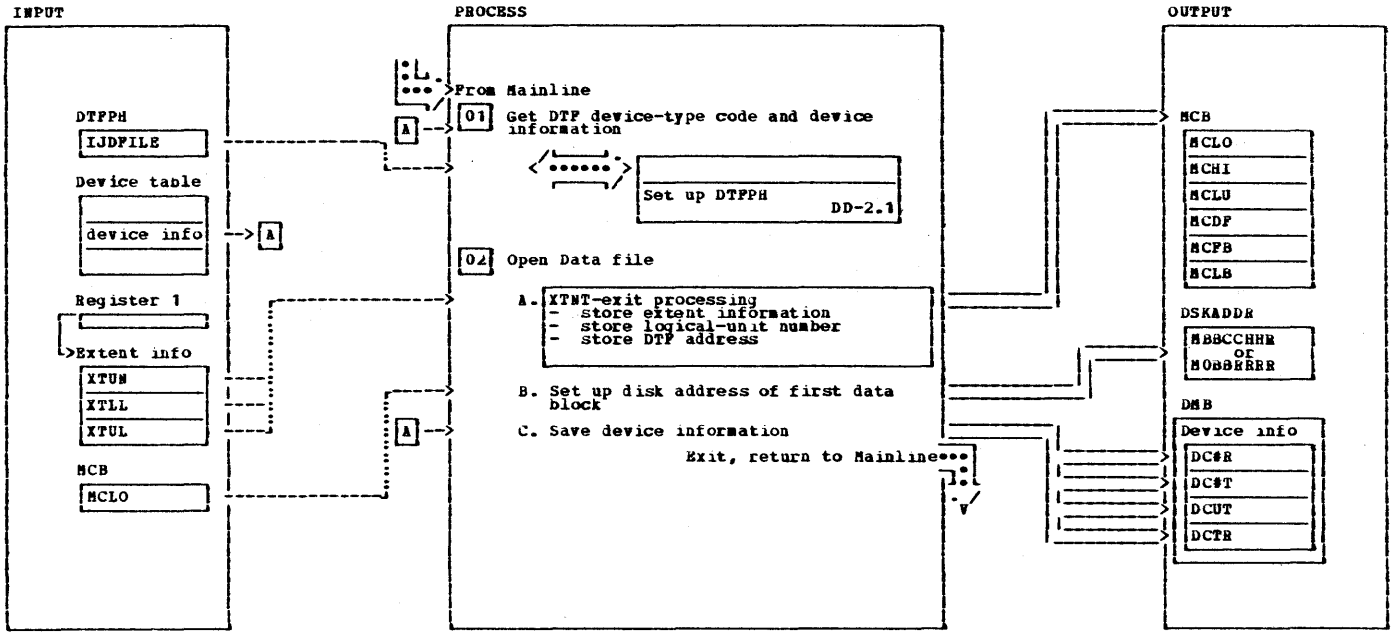
Read a record from disk (Subroutine)



NOTES	MODULE	LABEL	REF
This subroutine is used to do all I/O for the VSE/POWER files			
1 The 'M' of the disk request word is used to locate the associated module control block. The DTF pointer, stored in the module control block, is used to point to the CCB. The disk address is moved to the seek bucket which is pointed to by the seek CCW or locate CCW if FBA. The logical unit is then moved from the MCB to the CCB and also the record length is taken from the MCB and moved to the read CCW.		DD700	

NOTES	MODULE	LABEL	REF
2 The extent-description block is initiated with following values: - physical starting-block number - relative starting-block number - relative ending-block number			
2 The locate word is set up with following values, gotten from disk request word: - current block number - number of blocks to be processed			
3 Read the record and wait for I/O completion	EXCP WAIT	DD715	

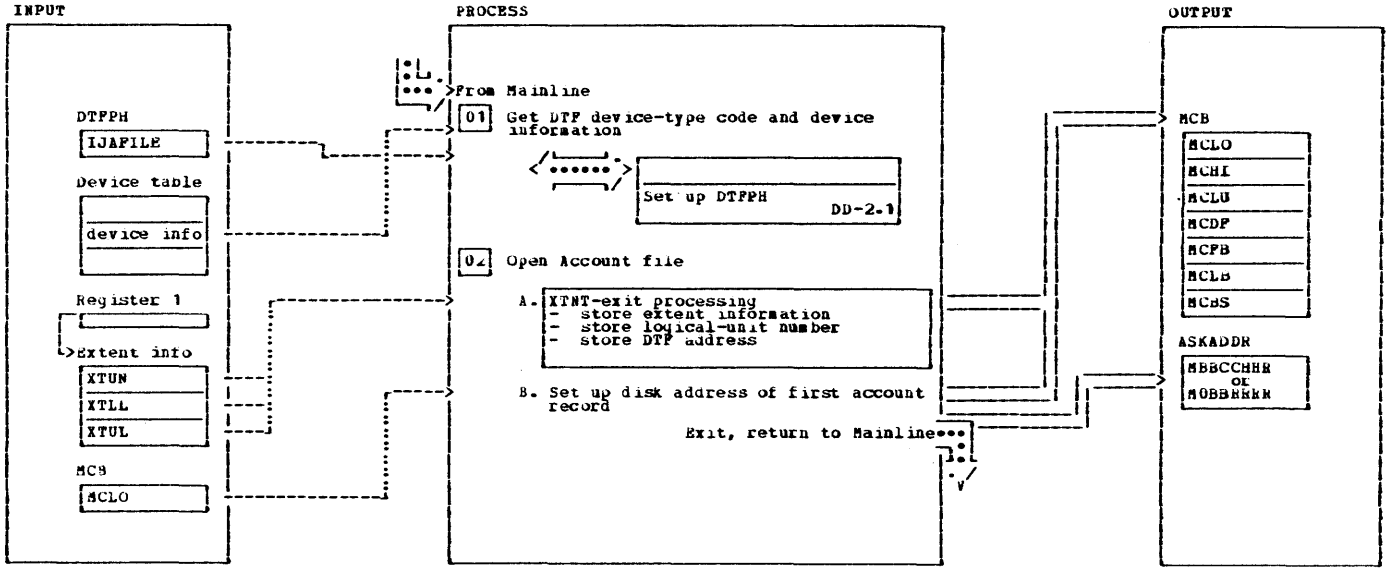
Open Data file



	NOTES	MODULE	LABEL	REF
	The Data file is opened for input operations. All relevant fields in the DMB and appropriate MCBs are initialized.			
1	The 'Set up DTFPH' routine returns the address of the device table entry associated with the device.		DD58	
2	The data file is opened.	OPENR		
2A	Control is given to this routine when OPEN is performed by IOCS. The logical-unit number and the DTF pointer are saved in the MCB. The extent information (lower & upper limit) is saved in the MCB.		DD71	
2A	If the Datafile resides on an FBA device, the relative ending block number is calculated for each extent. This is done by subtracting the physical starting-block number from the physical ending-block number and adding the relative starting-block number. (Note: the relative block number for the first extent is zero). Control is returned to the calling routine via the LBRET 1 macro intruction.	LBRET		

	NOTES	MODULE	LABEL	REF
2B	The disk address is set up to address the first data block. seek address is lower extent limit and record number is 1 for CKD; for FBA, block number is set to zero.			
2C	The number of records(DBLK) / track is calculated and saved in the DMB. Additionally the number of tracks/cylinder is obtained from the device-table entry and saved in the DMB. The number of track groups/cylinder is calculated to initialize the track group control table. Each track is represented by one byte containing head address + 1 of the corresponding track or, for the last-in-track-group or nonavailable track, x'00'.	GETVCE	DD60	

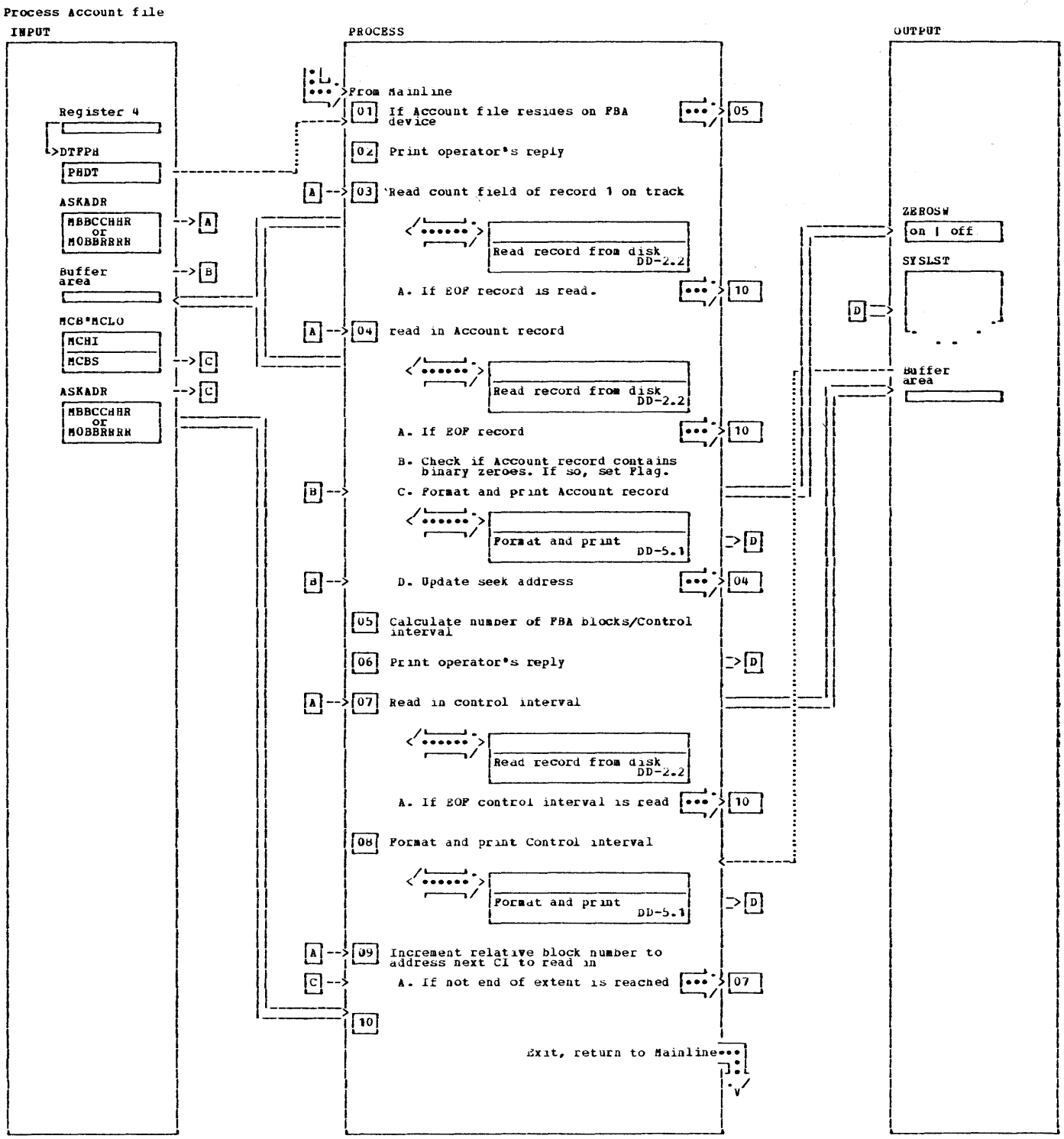
Open Account file



	NOTES	MODULE	LABEL	REF
	The Account file is opened for input operations. All relevant fields in the MCB are initialized.			
1	The 'Set up DTFPH' routine returns the address of the device table entry associated to the device.		DD72	
2	The Account file is opened by issuing the OPENR macro instruction	OPENR		
2A	Control is given to this routine when OPEN is performed by IOCS. The logical unit number and the DTF pointer are saved in the MCB. The extent information (lower & upper limit) are saved in the MCB.		DD70	
2A	If the Account file resides on an FBA device, the relative ending block number is calculated. This is done by subtracting the physical starting block number from the physical ending block number. (Note: the relative block number for the first extent is zero). Control is returned to the calling routine via the LBRET 1 macro intruction.	LBRET		

	NOTES	MODULE	LABEL	REF
2B	The lower limit of the extent is used to set up the disk address, the record number is set to zero in order to read the countfield of the first account record. If the Account file resides on an FBA device the block number is set to zero. The number of tracks/cylinder is saved in the MCB			

This page was left blank intentionally.

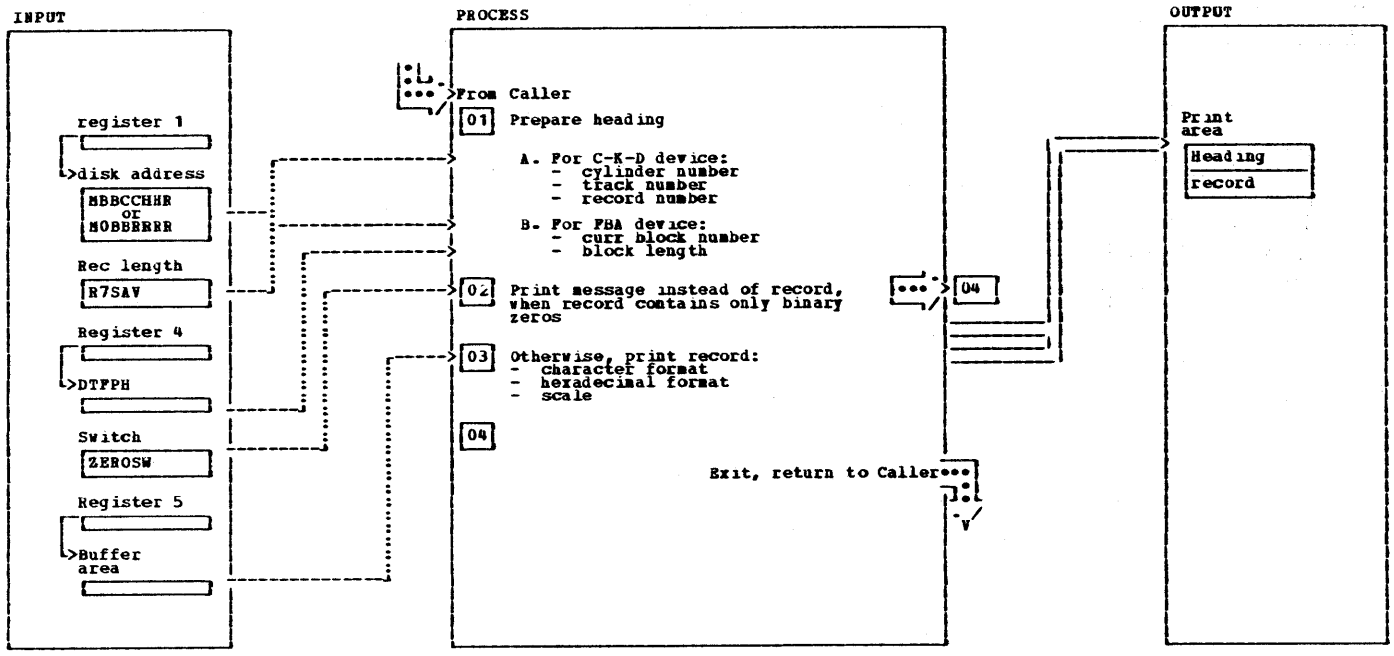

Process Account file
INPUT
PROCESS
OUTPUT
Register 4
DTFPH
PHDT
ASKADR
MBBCCHHR or MOBBRRRR
Buffer area
MCB*MCLO
MCHI
MCBS
ASKADR
MBBCCHHR or MOBBRRRR
From Mainline
01 If Account file resides on FBA device
02 Print operator's reply
03 Read count field of record 1 on track
Read record from disk DD-2.2
A. If EOF record is read.
04 read in Account record
Read record from disk DD-2.2
A. If EOF record
B. Check if Account record contains binary zeroes. If so, set Flag.
C. Format and print Account record
Format and print DD-5.1
D. Update seek address
05 Calculate number of FBA blocks/Control interval
06 Print operator's reply
07 Read in control interval
Read record from disk DD-2.2
A. If EOF control interval is read
08 Format and print Control interval
Format and print DD-5.1
09 Increment relative block number to address next CI to read in
A. If not end of extent is reached
10
Exit, return to Mainline
ZEROSW
on | off
SYSLST
Buffer area

Process Account file

NOTES	MODULE	LABEL	REF
1		DD74	
2 Message DUMP FUNCTION= reply is printed on a new page on SYSLST		SKIP	
3 The channel program is prepared to read in the count field of the first account record on the track. If an EOF record has been read (data length in count field = 0), a branch is taken to close the Files. Otherwise the buffer is cleared and the seek bucket is updated to read in record one.			
4D The seek address is incremented to address the next Account record to read in, unless the EOF record was read prior. If track change, update seek address to read in count field of the first Account record on the new track.			

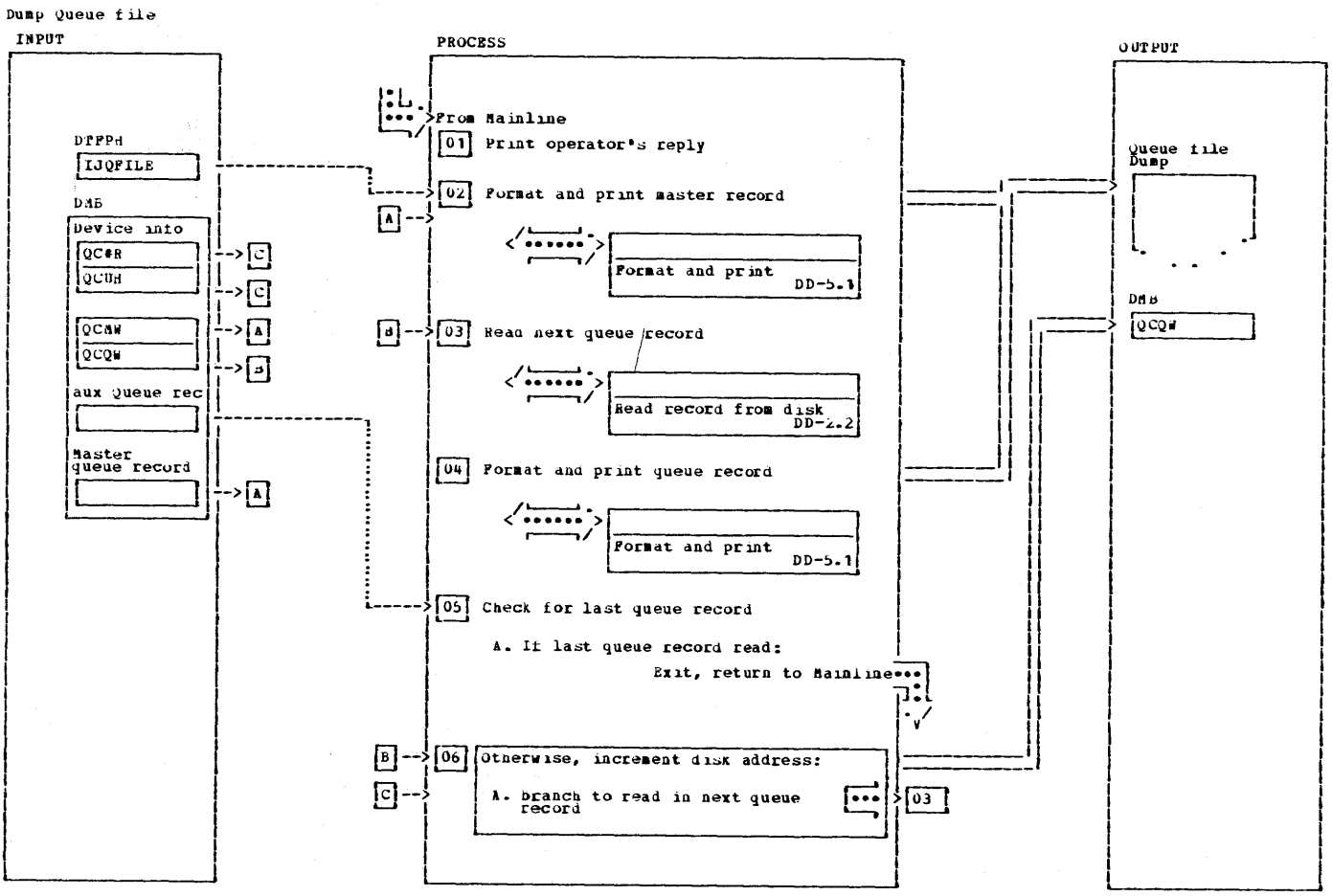
NOTES	MODULE	LABEL	REF
5 The number of FBA blocks occupied by on control interval is calculated. This is done by dividing the CI size (1995 bytes) by the physical FBA block size and rounding up the result. The CI size is 1995 bytes, regardless of the FBA block size		DD80	
6 Message DUMP FUNCTION= reply is printed on a new page on SYSLST		SKIP	
9 The relative block number is increased by the number of FBA blocks occupied by one Control interval			

Format and print (Subroutine)



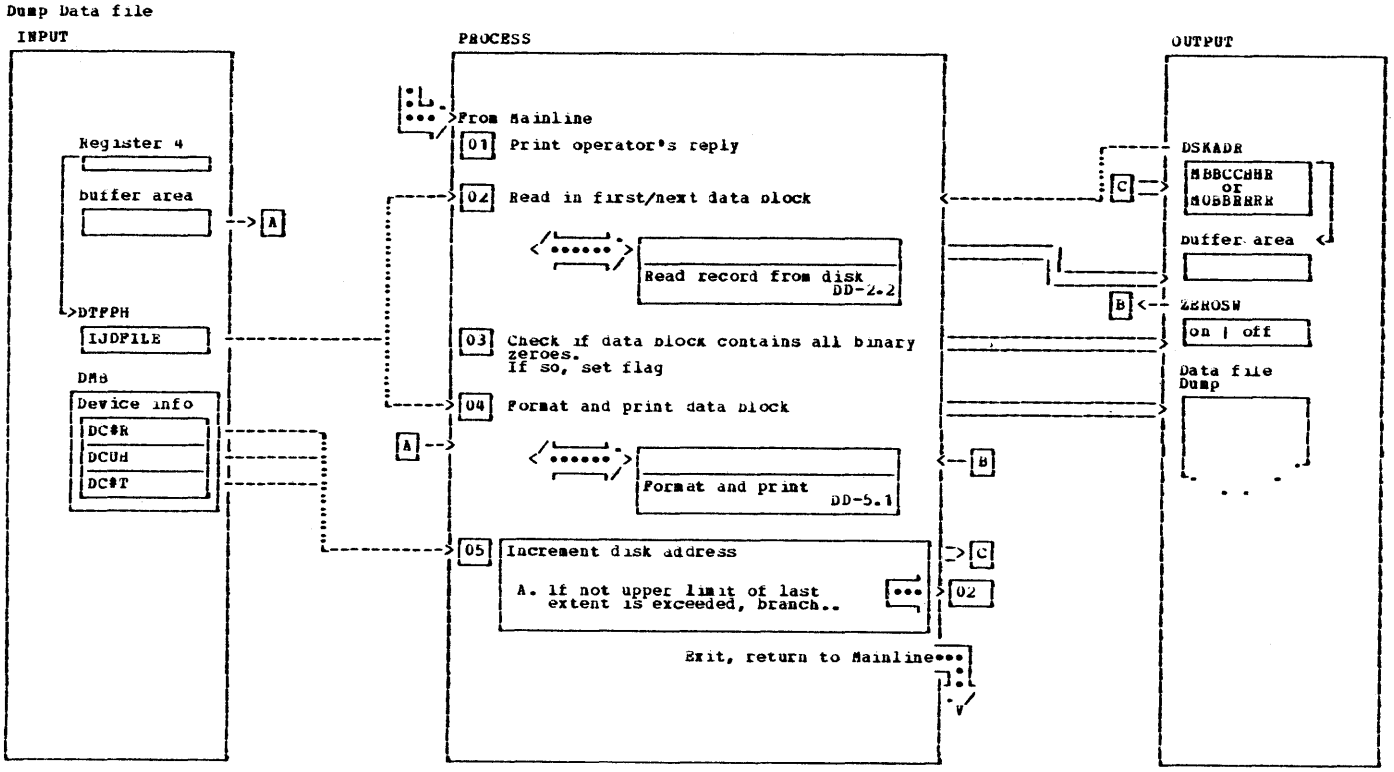
NOTES	MODULE	LABEL	REF
Each record is divided into segments of 100 bytes. For each segment, four lines are printed: line 1: the printable characters are printed. line 2: the zone part of the byte is printed. line 3: the numeric part of each byte is printed. line 4: a scale is printed to indicate the byte-sequence number. If a data record contains only binary zeros, a message is printed instead of the record.			

NOTES	MODULE	LABEL	REF
1		DSKDEBL	
1A The cylinder number, track number, record number and record length are converted to decimal and stored in the heading.			
1B The relative FBA block number and the block length are converted to decimal and stored in the heading of the first record.			
2 If the record contains all zeros, message THIS BLOCK/RECORD CONTAINS BINARY ZEROS ON DISK is issued.		DEBLOK	



NOTES	MODULE	LABEL	REF
This routine reads all queue records and prints them on SYSLST		DD300	
1 Message DUMP FUNCTION= reply is printed on a new page on SYSLST		SKIP	
3 The queue record is read in the auxiliary queue record area which is part of the DMB.		DD310	
5 The last queue record is indicated by 'D' as queue-record identifier			

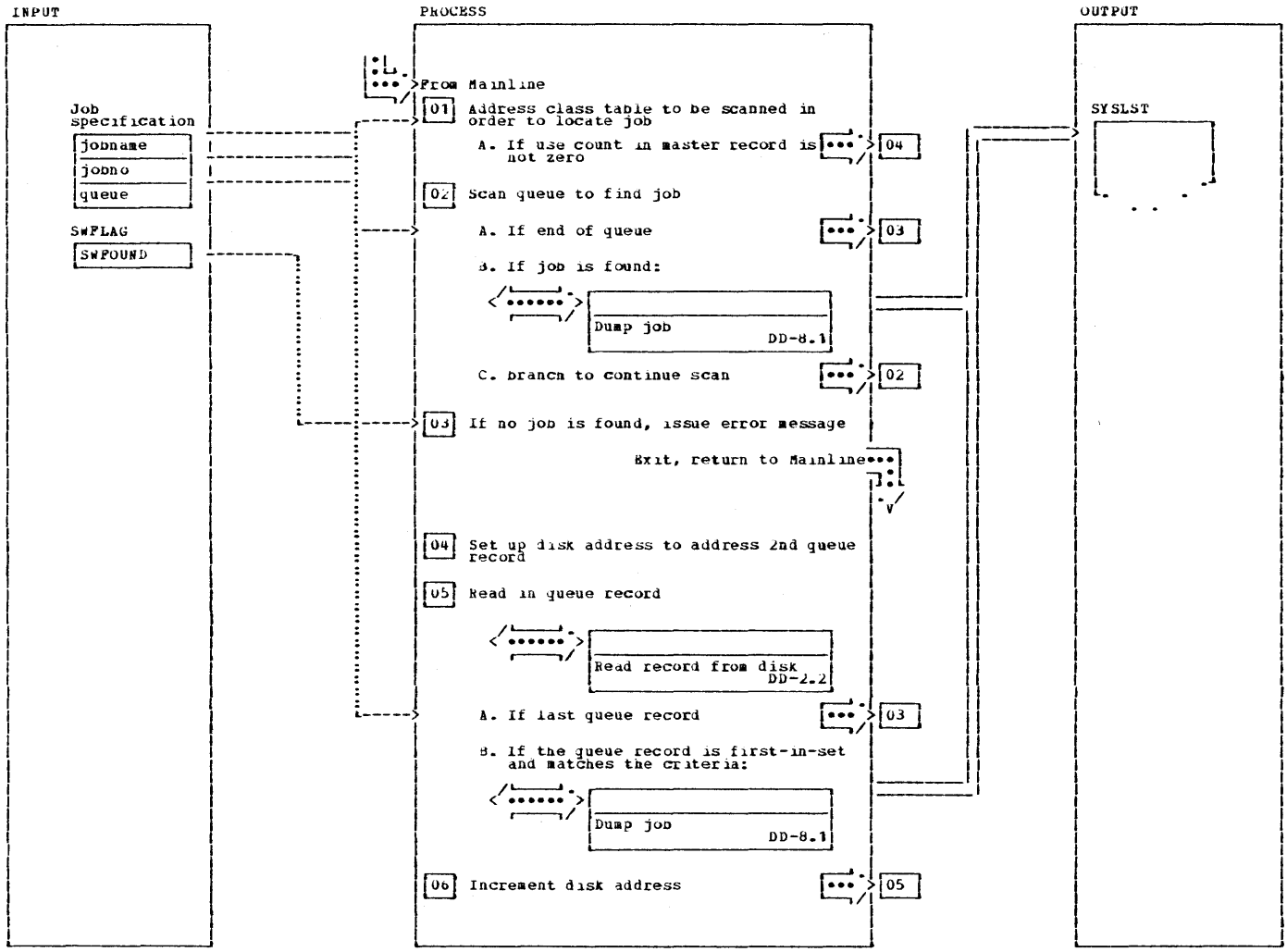
NOTES	MODULE	LABEL	REF
6 If FBA device, update curent block number by one. If CKD, increase current record number. If the new record number is higher than the maximum number of records per track, the current record number is set to one and the current track number is increased by one. If the new track number is higher than the number of tracks/cylinder, the track number is set to one and the cylinder number is increased by one.		QRUPDT	



NOTES	MODULE	LABEL	REF
This routine reads all data blocks and prints them on SYSLST		DD400	
1 Message DU8P FUNCTION= reply is printed on a new page on SYSLST		SKIP	
2 The data block is read in from the buffer area that is addressed by the disk address word			
5 For FBA, the current block number is incremented by the unit of transfer (that is the number of FBA blocks which are occupied by one data block).			
For CKD, the current record number is increased by one. If the new record number is higher than the maximum number of records per track, the current record number is set to one and the current track number is increased by one. If the new track number is higher than the number of tracks/cylinder, the track number is set to one and the cylinder number is increased by one.		DBUPDT	

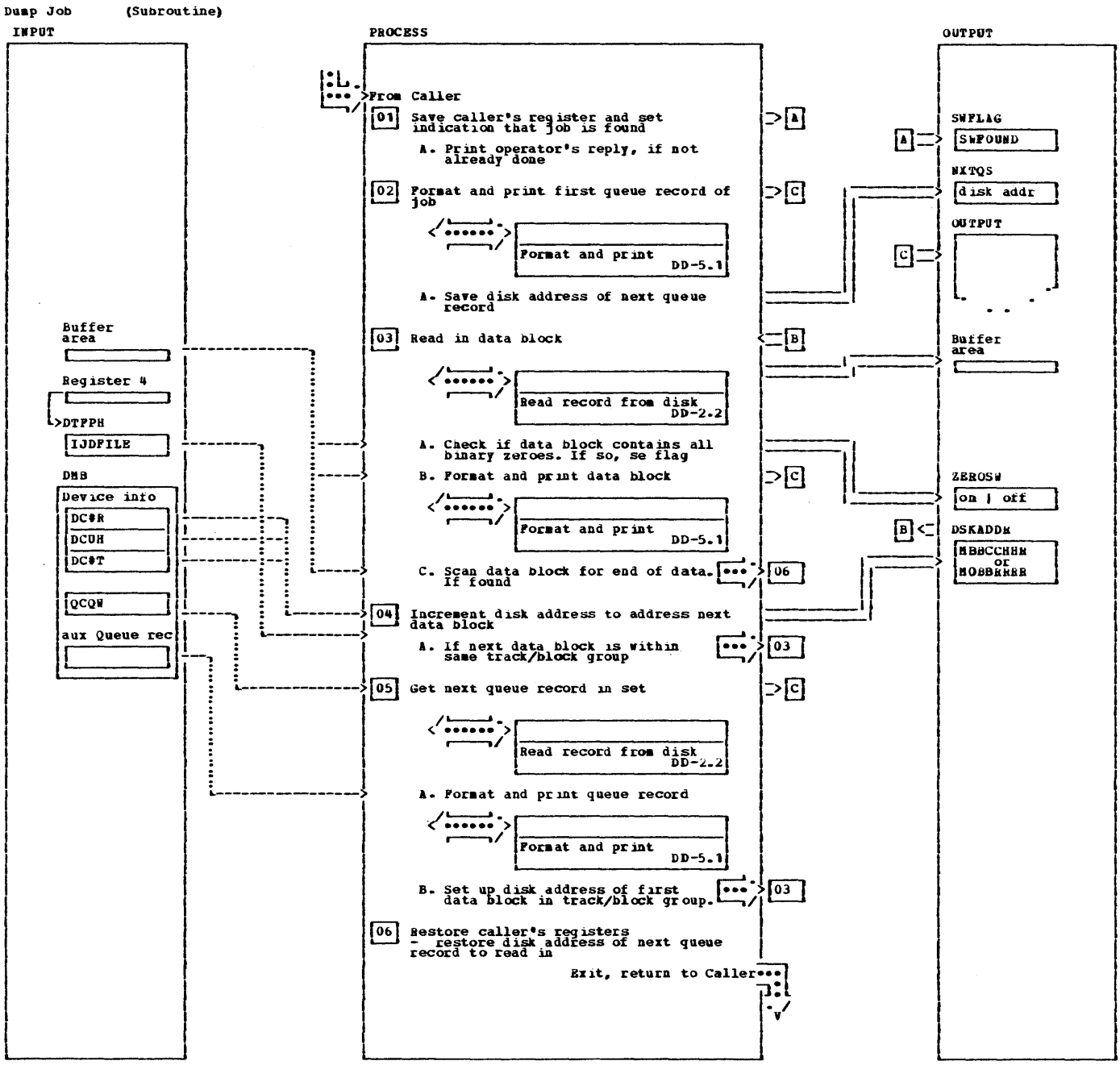
NOTES	MODULE	LABEL	REF
If this data block is already partially outside of the extent, the module index is increased by four to address the next data-file extent. If there is no next data-file extent, meaning that already all data blocks are printed out, return to the caller is made with a displacement to finish up data-file-dump processing. Otherwise the disk address is updated with a lower-extent limit of the new data-file extent.			

Dump specific job



NOTES	MODULE	LABEL	REF
1 The class table, placed in the master record, is addressed and the pointer to the queue (either RDR, LST or PUN) is set depending on user's specification.			
1A If the use count is zero, it indicates that the last VSE/POWER system terminated normally and that we can rely on the contents of the class table. Otherwise we have to scan the entire queue file in order to locate the job we are looking for.			

NOTES	MODULE	LABEL	REF
3 JOB ENTRY NOT FOUND is issued.			
4 The lower extent limit (CCHH) is used to initialize the seek bucket. If the queue file resides on an FBA device, the relative block number is set to one. Note: the first queue record is used for internal purposes only.			
5A The last queue record is identified by an ID of 'D'			
6 The disk adress of the queue record is incremented to address the next queue record to read in.			


Dump Job (Subroutine)
INPUT
PROCESS
OUTPUT
From Caller
01 Save caller's register and set indication that job is found
A. Print operator's reply, if not already done
02 Format and print first queue record of job
Format and print DD-5.1
A. Save disk address of next queue record
03 Read in data block
Read record from disk DD-2.2
A. Check if data block contains all binary zeroes. If so, se flag
B. Format and print data block
Format and print DD-5.1
C. Scan data block for end of data. If found
04 Increment disk address to address next data block
A. If next data block is within same track/block group
05 Get next queue record in set
Read record from disk DD-2.2
A. Format and print queue record
Format and print DD-5.1
B. Set up disk address of first data block in track/block group.
06 Restore caller's registers - restore disk address of next queue record to read in
Exit, return to Caller
Buffer area
Register 4
DTFPH
IJDFILE
DMB
Device info
DC$R
DCUH
DC$T
QCQW
aux Queue rec
SWFLAG
SwFOUND
NXTQS
disk addr
OUTPUT
Buffer area
ZEROSW
on | off
DSKADDR
MBBCCHHR or MOBBRRRR

Dump Job (Subroutine)

NOTES	MODULE	LABEL	REF
This subroutine prints out all data blocks and associated queue records allocated by a job on SYSLST		DUMPJOB	
1 The operator's answer is printed on a new page on SYSLST			
2 The first-in-set queue record is formatted and printed			
2A Since the queue set may consists of multiple queue records, the disk address of the next queue record in the set or the of the next queue record in the file in case of abnormal termination is saved.			
3		DJOB10	
3C Each record in the data block is examined for the end-of-data record			
4 If the data file resides on an FBA device, the current block number is incremented by the unit of transfer (that is the number of FBA blocks which are occupied by one data block). The new block number is then checked against the block-group-size value. If the new data block does not fit entirely in the blockgroup, the next queue record in the set is read in.			

NOTES	MODULE	LABEL	REF
If the data file resides on a CKD device, the current record number is incresed by one and a check is made if the new data block fits in the current track. If the new data block is on the next track, the track number is incremented by one and the record number is set to one. A check is made if the new track exceeds the track group. If so, the next queue record is read in.			
5 The address of the next queue record is obtained from the current queue record.			
6 the disk address of the next queue record in the class chain or the next queue record in the queue file is restored.			

IPW$$DQ - VSE/POWER Delete Queue Set from Chain	
Label	Routine
DQ00	Function Entry
DQ08	Examine Queue Record
DQ32	Delete First in Class
DQ36	Delete Middle in Class
DQ40	Delete Last in Chain
DQ44	Delete Complete Chain
DQ50	Function Exit
DQ90	Address Conversion

Services Used	
Service	Macro
Resource Management	IPW$RLR IPW$RSR
Disk / Tape Service	IPW$CTT IPW$RDQ IPW$WTQ

Called by	
Module	Description
IPW$$CA	Command Processor (PALTER)
IPW$$CL	Command Processor (PDELETE)
IPW$$LR	Logical Reader
IPW$$LW	Logical Writer
IPW$$NR	Network Receiver
IPW$$NT	Network Transmitter
IPW$$OF	Offload Queues
IPW$$TR	Task Terminator
IPW$$XR	Execution Reader

Labels	Chart DQ: IPW$$DQ - Delete Queue Set from Chain		Modified Data Fields	Reg. Usage	Calls
DQCS	The first 16 bytes constitute the section descriptor:				
	'DQCS 10 release'				
DQ00	Function Entry				
	Save caller registers 14 through 9 inclusive.		IPW$DSV		
	Set addressability for queue record area.			R5	
	If no tape spooling, branch to......>	DQ01			
	Read in the trailing queue record.				IPW$CTT
	Branch to..........................>	DQ55			
DQ01	Set function track indicator to D (delete in progress).		TCFT(IPW$DTC)		
	Set addressability for disk management block (DMB).			R6	
	Lock DMB.				IPW$RSR
	Address first-in-set.		TCQW(IPW$DCT)		
DQ08	Save "flush hold" indicator from current queue record.			R4	
	Save remaining copy count from current queue record.			R4	
	Save current copy group index.			R4	
	Save remaining restart page count from current queue record. (It still exists if PSTOP, RESTART was entered).			R8	
	Examine Queue Record				
	Read first-in-set queue record.				IPW$RDQ
	If the previous class pointer addresses the record itself, the queue set was not yet added to a class chain........................>	DQ52			
	If function byte indicator 'hold', the queue record execution switch is reset and the shared system id is set to X'00'. The queue record is rewritten,.........................>	DQ52	TCCB(IPW$DTC) QRXS(IPW$DQR) QRSI(IPW$DQR)		IPW$WTQ
DQ10	If request from command processor...>	DQ20			
	If 'KEEP' disposition, branch to....>	DQ12			
	If not, 'FLUSH,HOLD', branch to.....>	DQ20			
	Set 'HOLD' disposition.		QRDP(IPW$DQR)		
DQ13	Reset saved flush hold indicator....>	DQ16	QRDI(IPW$DQR)		
DQ12	Set 'LEAVE' disposition.		QRDP(IPW$DQR)		
	If 'FLUSH,HOLD', branch to..........>	DQ13			R4

Labels	Chart DQ: IPW$$DQ - Delete Queue Set from Chain		Modified Data Fields	Reg. Usage	Calls
DQ16	Reset execution switch.		QRXS(IPW$DQR)		
	Set saved remaining restart page count.		QRRR(IPW$DQR)	R8	
	Reset system-id indicator.		QRSY(IPW$DQR)		
	Rewrite the record.				IPW$WTQ
	Exit.....>	DQ52			
DQ20	The address of the correct class table index entry is calculated		QRSI(IPW$DQR) QRQI(IPW$DQR)		
DQ26	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$RDQ
	Zero its previous class pointer.		QCQP(IPW$DQC)		
	Rewrite this record.				IPW$WTQ
	Store new forward pointer in class table entry.		CTQF(MCT)		
	Exit.....>	DQ50			
DQ36	Delete Middle-in-Class Set				
	Read previous and next first-in-set queue records in turn.				IPW$RDQ
	Exchange class pointers.				
	previous record - forward pointer		QCQN(IPW$DQC)		
	next record - previous pointer		QCQP(IPW$DQC)		
	Write the records back in turn.				IPW$WTQ
	Exit.....>	DQ50			
DQ40	Delete Last-in-Class Set				
	Read previous first-in-set queue record.				IPW$RDQ
	Zero its next class pointer.		QCQN(IPW$DQC)		IPW$WTQ
	Rewrite this record.				
	Store new backward pointer in class table entry.		CTQL(MCT)		
	Set live bit (class ECB).		CTQL(MCT)		
	Exit.....>	DQ50			

Labels	Chart DQ: IPW$$DQ - Delete Queue Set from Chain		Modified Data Fields	Reg. Usage	Calls
DQ44	Delete Complete Chain				
	Zero forward and backward pointers in the appropriate class table entry.		CTQF(MCT) CTQL(MCT)		
	If the spool files are not shared, examine next class.....>	DQ50			
	The appropriate bit in the sysid class table is turned off for all sysids, because there are no longer any entries in the class list.				
	Examine next class.....>	DQ50			
	Function Exit				
DQ50	Set previous set pointer. Reset execution switch. Reset system-id indicator. Rewrite the record.		QRQP(IPW$DQR) QRXS(IPW$DQR) QRSY(IPW$DQR)		IPW$WTQ
DQ52	Set function track indicator to C'U' (unchained).		TCFT(IPW$DTC)		
	If it is command-processor task or save-account function.....>	DQ55			
	Unlock DMB.				IPW$RLR
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	
	For FBA devices it is not necessary to convert from absolute to relative, because READ/WRITE will be done with relative block number only.				
	If FBA device is used, store absolute block number (register 2) in register 1 (relative block number).				
	Return to caller via register 14.			R14	

IPW$$ER - VSE/POWER 3540 Diskette Reader	
Label	**Routine**
ERCS	Routine Entry
PS00	PUB Scan Routine
OP00	Open Extent Routine
IP00	Initialize Physical Data Area
RD00	Read Data Block
DB00	Deblock Routine
EX00	Exit Routine
SR10	Subroutines

Services Used	
Service	**Macro**
Task Management	IPW$WFC
Storage Management	IPW$RLW IPW$RSW
Message Service	IPW$GAM
Notify User Message	IPW$NTY

Function used	
Module	**Macro**
IPW$$LW	IPW$PLR - Put Logical Record
IPW$$NU	IPW$OLI - Open Logical Interface
IPW$$OE	IPW$OEF - Open Diskette File
IPW$$LU	IPW$ULP - Update LUB/PUB Table

Called by	
Module	**Description**
IPW$$CS	Command processor (PSTART)
IPW$$LR	Logical Reader

Labels	Chart ER: 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 release'				
	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 - **** - Work register			R5	
	6 - **** - Pointer to last CCW in string			R6	
	7 - **** - Record counter per track			R7	
	8 - IPW$DPW - Physical work space			R8	
	9 - PRCS - Physical reader base register			R9	
	10 - IPW$DPA - VSE/POWER nucleus			R10	
	11 - IPW$DTC - Task control block			R11	
	12 - **** - Reserved for nucleus use			R12	
	13 - IPW$DSV - Task save area			R13	
	14 - **** - Subroutine linkage register			R14	
	15 - **** - Subroutine base register			R15	
	Routine Entry				
RE00	If this routine is entered from the the Logical Reader with a connected 3540 diskette device (alternate diskette processing LWER=X'01'), branch to.....>	OP00			
	If this routine is entered from the the Logical Reader in case of either primary diskette process (LWER=X'08') or dynamic diskette process (LWER=X'80') branch to.....>	RE10			
	Otherwise this routine is entered on behalf of pstart command for a 3540 diskette reader Save the parameter values which were passed in register 1 in register 6.			R6	
	Open the interface to the Logical Reader routine.				IPW$OLI
	Reserve 3540 physical work space to hold 3540 diskette information.				IPW$RSW
	Save the real address of the physical work space.		PERA(IPW$DPW)		
	Anchor the 3540 PWS in TCB.		TC3W		

Labels	Chart ER: 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)		
	Indicate primary diskette process in the TCB.		LWER		
RE10	Load the address of the master record register 3.			R3	
	Clear the option FEED.		PEOP(IPW$DPW)		
	If the option FEED in the master record is not on, then branch to... >	RE20			
	Turn on the option FEED in the 3540 physical work space.		PEOP(IPW$DPW)		
RE20	Check if there is already a 3540 device assigned. That is the case on primary process at 1st occurence, CP made the assignment, or 2nd RDR stmt in one jobstream, the pubscan and assign function was performed already In this case skip to.....>	OP00			

Labels	Chart ER: IPW$$ER - 3540 Diskette Reader		Modified Data Fields	Reg. Usage	Calls
	Pub Scan Routine				
PS00	Establish addressability to PUB table			R4	
PS05	Scan PUB table for a 3540 device (T3540) and if found.....>	PS15			
PS10	Get next PUB addressed and check if end of table reached. If not.....>	PS05			
	Otherwise no 3540 device is defined in pub, device is down or in use and therefore to issue msg branch to....>	PS30			
PS15	A 3540 device was found. Convert binary format of cuu in character format of cuu into PWS.		PEDW		
	Setup 'LOCATE FREE PUB' request On return, if rc ¬0 continue.....>	PS10			IPW$ULP
	Setup 'LOCATE AND ASSIGN LUB' request On return, if rc ¬0,				IPW$ULP
	Setup 'UNASSGN RELATED LUB' request and start over again.....>	PS10			IPW$ULP
PS20	Get logical unit number and branch to test if device operational.....>	SR00			
	On return if device not operational Setup 'UNASSGN RELATED LUB' request to continue scan branch to.....>	PS10			IPW$ULP
	Otherwise branch to open the just assigned 3540 device.....>	OP00			
PS30	Issue message : 1Q90I * $$ RDR STATEMENT NOT PROCESSED, JOB FLUSHED to the submitter of the job, which can be rje task or spool management.				IPW$GAM IPW$NTY
PSEX	Set flush stop code in TCB and branch to exit routine.....>	EX00	TCFT		

Labels	Chart ER: IPW$$ER - 3540 Diskette Reader		Modified Data Fields	Reg. Usage	Calls
	Open Extent				
OP00	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)		
OP05	Open the first 3540 volume.				IPW$OEF
	Check the open return code:				
	If open successful (PEOC=C'0'), branch to..... >	OP10			
	If forced end of volume (PEOC=C'E'), branch to..... >	FD00			
	Check the 3540 communication byte for primary diskette processing (LWER=X'08'), and if not branch to..>	OP07			
	Check if high level 3540-PWS present if not (PEHA=0), branch to.....>	EX30			
OP07	Set the termination byte in the TCB to C'F' to indicate flush, and branch to flush..... >	EX00	TCTT(IPW$DTC)		
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)		

Labels	Chart ER: IPW$$ER - 3540 Diskette Reader		Modified Data Fields	Reg. Usage	Calls
OP30	Initialize the pre-SEEK address to seek for record 25 on the next track.		PE3O(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$RSW
	Save the CCB pointer in the physical work space.		PECO(IPW$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$DPW)	R2	
	The size of the second physical data buffer to be reserved is calculated in register 5 and divided by the record length to test if it contains an integral number of data records. If the remainder in register 4 is zero, the data buffers to be allocated do not cross page boundaries, and a branch is made to >	IP10		R2,R4 R5	
	Otherwise, increase the number in register 1 by one to allocate one buffer more in the second data area.			R1	
IP20	The number of data buffers to be allocated in the second data area is saved in the physical work space.		PENN(IPW$DPW)		
	Calculate the total space required for the second data area in register 5 (number of buffers multiplied by record length).			R4,R5	

Labels	Chart ER: IPW$$ER - 3540 Diskette Reader		Modified Data Fields	Reg. Usage	Calls
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$RSW
	If the space is to be used as a second data area, branch to..... >	IP30			
	Otherwise, save the virtual address of the space to be used as a single data area in the physical work space.		PECV(IPW$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)		
	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	

Labels	Chart ER: IPW$$ER - 3540 Diskette Reader		Modified Data Fields	Reg. Usage	Calls
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$DPW)	R7	
RD10	Store the current record number in the physical work space.		PESK(IPW$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.				

Labels	Chart ER: IPW$$ER - 3540 Diskette Reader		Modified Data Fields	Reg. Usage	Calls
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$WFC
	Restore the address of the first CCW.		CBCA(IPW$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			
	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			

Labels	Chart ER: IPW$$ER - 3540 Diskette Reader		Modified Data Fields	Reg. Usage	Calls
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$DTC)		
DB55	Branch and link to pass the record to the logical reader ..... >	SR10		R5	
	If the last record on the last cylinder has not yet been passed to the logical reader, branch to continue processing..... >	DB00			
	Branch and link to reset a possible inserted pre-SEEK CCW to a read CCW >	SR20		R5	
DB60	Release the first or only part of physical data buffer space.				IPW$RLW
	If a second data area has not been reserved, branch to clear the physical work space..... >	DB70			
	Release the second physical data area.				IPW$RLW
DB70	Set the buffer number to zero.		PENN(IPW$DPW)		
	Check if there is a continuation diskette. If so, branch to open the next extent..... >	OP00			

Labels	Chart ER: IPW$$ER - 3540 Diskette Reader		Modified Data Fields	Reg. Usage	Calls
	Exit Routine				
FD00	If the option FEED in the PWS is not on, branch to..... >	EX00			
	Branch and link to feed the next diskette..... >	SR50			
EX00	Check for data file processing, and if so, branch to handle end of data >	EX40			
	Check if the SYSIN file is linked to a card reader (alternate diskette process) or if dynamic diskette process, and if so, branch to prepare for resetting the linkage..... >	EX60			
	Otherwise it is primary diskette processing. In that case at first check for job boundary, and if so, do not insert the EOF record .....>	EX10			
	Load the address of the EOJ record into register 0 and the length into register 1.			R0,R1	
	Branch and link to pass the EOJ record to the logical reader..... >	SR10		R5	
	Set the disposition indicator in the queue record to C'H' to indicate job in HOLD state.		QRDP(IPW$DQR)		
	Issue message '1089I cuu EOJ ADDED jobname jobnumber'				IPW$GAM
EX10	If high level 3540-PWS present.....>	EX60			
EX20	Branch and link to send the unit exception signal to the logical reader..... >	SR10		R5	
EX30	Set the stop condition in the TCB.		TCTT(IPW$DTC)		
	Exit to the terminator routine (IPW$$TR), which results in a detach of the current 3540 diskette task.				IPW$$TR
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
	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.				

Labels	Chart ER: IPW$$ER - 3540 Diskette Reader		Modified Data Fields	Reg. Usage	Calls
	Subroutines				
SR10	Pass the record to the logical reader.				IPW$PLR
	Return to caller via link register 5.			R5	
SR20	Load the address of the forced pre-SEEK CCW into register 2.			R2	
	If there is no forced pre-SEEK CCW, branch to return..... >	SR25			
	Make the address virtual and use it as a base register.			R2	
	Move the read CCW saved in the physical work space back into the CCW string.		CWDS(IPW$DCW)		
	Clear the forced pre-SEEK CCW address in the physical work space.		PEBS(IPW$DPW)		
SR25	Return to caller via link register 4.			R4	
	Subroutine to insert a pre-SEEK CCW in the middle of the CCW string to read part of a track				
SR40	If a short string has to be executed which is not on the last track, branch to..... >	SR45			
	Otherwise, change the pre-SEEK address to the current seek address, and calculate the number of remaining records to be read in register 1, depending on the last record of the extent.		PESO(IPW$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.		PEBS(IPW$DPW)		
	Return to caller via link register 14.			R14	

Labels	Chart ER: IPW$$ER - 3540 Diskette Reader	Modified Data Fields	Reg. Usage	Calls
	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.			
	Subroutine to Build a CCB with CCW to SENSE			
SR60	Reserve work space			IPW$RSW
	Calculate the displacement (virtual address and real address).		R2	
	Build the CCB. Set : SENSE INFO DESIRED.	CBPI		
	Build the CCW SENSE.			
	Execute channel program.			SVC0
	Wait for operation to complete. Copy the CCB status byte in PWS	PESD		IPW$WFC
	Release work space.			IPW$RLW
	Return to caller.			

IPW$$FQ - VSE/POWER Free Queue Set Storage	
Label	Routine
FQ00	Function Entry
	Examine Queue Record
	Update Free Queue
FQ52	Function Exit

Services Used	
Service	Macro
Resource Management	IPW$RLR
	IPW$RSR
Disk / Tape Service	IPW$RDQ
	IPW$WTQ

Called by	
Module	Description
IPW$$CL	Command Processor (PDELETE)
IPW$$LR	Logical Reader
IPW$$LW	Logical Writer
IPW$$NR	Network Receiver
IPW$$NT	Network Transmitter
IPW$$OF	Offload Queues
IPW$$TR	Task Terminator
IPW$$XR	Execution Reader

Labels	Chart FQ: IPW$$FQ - Free Queue Set Storage		Modified Data Fields	Reg. Usage	Calls
FQSD	The first 16 bytes constitute the section descriptor:				
	'FQCS release'				
	Function Entry				
FQ00	Save caller registers 14 through 9 inclusive.		SVDS(IPW$DSV)		
	If tape spooling for this task, exit to.....>	FQ05			
FQ02	Set track action indicator F		TCFT(IPW$DTC)		
	Set addressability for:				
	Queue record area (QRA)			R5	
	Disk management block (DMB)			R6	
	Lock DMB.				IPW$RSR
	Examine Queue Record				
	If the request is from CP or from a reader task.....>	FQ03			
	If the disposition is 'L' or 'H'....>	FQ00			
FQ03	Put address of first-in-set in TCB		TCQW(IPW$DTC)		
FQ04	If disk address of first-in-set is equal to the previous set pointer then the set has never been added to any chain.....>	FQ50			
FQ07	Read back first-in-set queue record.				IPW$RDQ
	Update Free Queue				
FQ50	If queue-record traps are active (UPSI 1), check that a delete queue-set was done before the free queue-set. If not, force program check....>	FQSD	QCCF(IPW$DQC) QCBF(IPW$DQC) QCQI(IPW$DQC)		
	Clear auxiliary queue record area. Set free record identifier.				
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
	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

Labels	Chart FQ: IPW$$FQ - Free Queue Set Storage	Modified Data Fields	Reg. Usage	Calls
	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 if not the CP task or the save-account task.			IPW$RLR
FQ65	Restore caller registers and return.		R14-R9	IPW$RET

IPW$$GA - VSE/POWER Get Account Record	
Label	**Routine**
GACS	Function Entry
GAOP	Open Account File in Get Mode
GAGT	Read Account Record
GAER	Prepare for Erase Only Close of Account File
GAKP	Prepare for Keep Close of Account File
GACL	Close Account File Get Mode, Restore Put Mode
GAEX	Function Exit
GAWM	Restore Write Mode Channel Program Subroutine

Services Used	
Service	**Macro**
Task Management	IPW$WFC
Resource Management	IPW$RLR IPW$RSR
Storage Management	IPW$RLW IPW$RSW

Called by	
Module	**Description**
IPW$$I5	Initialize Account File
IPW$$SA	Save Account
IPW$$TR	Task Termination

Labels	Chart GA: IPW$$GA - Get Account Record		Modified Data Fields	Reg. Usage	Calls
GASD	The first 16 bytes constitute the Section Descriptor:				
	'GACS release'				
	On exit, the following register contents are relevant:				
	0: Account record length			R0	
	1: Virtual address account record area			R1	
	The following registers will be used by IPW$$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$OAF macro.				
	Open functions are:				
	• Signal OPEN running to TCB. • Reserve ACB, • Reserve real I/O buffer space, • Set ACB from (normal) PUT mode to GET mode.		FCAT(IPW$DTC)		
	Caller's registers are saved.				IPW$SAV
	Address of ACB is loaded in register 5.			R5	
	Address of DMB is loaded in register 6.			R6	
	The ACB and the DMB are reserved.				IPW$RSR

Labels	Chart GA: IPW$$GA - Get Account Record		Modified Data Fields	Reg. Usage	Calls
	Register 1 is loaded with the maximum record size.				
	Work space is reserved using register 1 as parameter register.				IPW$RSW
	If work space is available (register 0 = nonzero), branch.....>	GA01			
	Otherwise, indicate immediate stop in TCB for save account function, and reset caller into active status to service terminator routine, branch .>	GAC7	TCTT(IPW$DTC) TCAT(IPW$DTC)		
GA01	The (virtual) buffer space address is saved in the ACB.		ACWA(IPW$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)		
	The account file is now ready to be accessed in GET mode.		TCAT(IPW$DTC)		
	A branch is made to return to the caller.....>	GAEX			

Labels	Chart GA: IPW$$GA - Get Account Record		Modified Data Fields	Reg. Usage	Calls
GAGT	This routine is entered if IPW$$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$SAV
	The address of the ACB is loaded in register 5.			R5	
	The address of the DMB is loaded in register 6.			R6	
	The address of the buffer space is loaded in register 8.			R8	
	The caller's register 0 save area is initialized to zero (EOF).		SVR0(IPW$DSV)		
	Head and record count in seek address are copied from the count field.		ACSA(IPW$DAC)		
	If not yet end of extent or end of cylinder, branch.....>	GAG1		R7	
	If not end of cylinder, (so end of extent), branch.....>	GAG3			
	If upper limit reached, branch.....>	GAG6			
	Otherwise, using register 1 as a work register, cylinder number in seek address is incremented by one, and head and record number are set to zero.		ACSA(IPW$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 GA: IPW$$GA - Get Account Record		Modified Data Fields	Reg. Usage	Calls
GAG2	The multitrack bit is set to zero.		ACRW(IPW$DAC)		
GAG3	Sector value next cycle is set.		ACSE(IPW$DAC)		
	EXCP parameter register 1 is loaded with the account file CCB address.			R1	
	An EXCP is issued to read the next account record.				EXCP
	An IPW$WFC is issued to wait for I/O completion.				IPW$WFC
	On completion, if an EOF record was read, branch.....>	GAG6			
GAG5	The account record block length (logical record length + 8) is stored in the caller's register 0 save area.		SVR0(IPW$DSV)		
GAG6	The virtual address of the account record is stored in the caller's register 1 save area.		SVR1(IPW$DSV)		
	Reset caller to active status to service terminator routine.		TCAT(IPW$DTC)		
	A branch is made to return to the caller.....>	GAEX			
GAER	This routine is entered if IPW$$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.				IPW$SAV
	Register 7 is set up to point to the KEEP function indicator.			R7	
	A branch is made to continue.....>	GAC0			

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

Labels	Chart GA: 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.			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$DAC)		
	Register 1 is loaded with the address of the account CCB.			R1	
	An EXCP is issued to write an EOF record.				EXCP
	The task is put in a wait state until I/O completion.				IPW$WFC
	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)		

Labels	Chart GA: IPW$$GA - Get Account Record		Modified Data Fields	Reg. Usage	Calls
	Current seek addresses in ACB are initialized at the extent lower limit.		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$DAC)	R1	
	ACB update is now complete, therefore the account ECB is posted.		ACEB(IPW$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

Labels	Chart GA: IPW$$GA - Get Account Record		Modified Data Fields	Reg. Usage	Calls
GAC7	The ACB and DMB are released.				IPW$RLR
GAEX	This routine is the common IPW$$GA exit routine to the caller.				
	The caller's registers are restored, and control is returned to the caller.				IPW$RET
				R14	
GAWM	This subroutine resets ACB and account channel program to PUT mode.				
	If the ACB is found to be in PUT mode, a branch is made to bypass ACB reset.....>	GAW1			
	Otherwise, the multitrack indicator, which may have been switched off by the IPW$GAR function, is set ON.		ACRW(IPW$DAC)	R4	
	The account channel program is set to write by swapping read and write channel programs.		ACRW(IPW$DAC) ACPM(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	

IPW$$GD - VSE/POWER Get Data Record	
Label	**Routine**
GD00	Function Entry
GD02	Get Next Tape Block
GD10	Get Next Disk Block, Single Buffering
GD10	Get Next Disk Block, Double Buffering
GD43	Increment Disk Address
GX00	Extended Record Processing
GDRT	Function Exit
GD60	Get Next Track Group Subroutine
GD70	Check Next Data Block Address Subroutine
GD80	Increment Disk Address Subroutine

Services Used	
Service	**Macro**
Storage Management	IPW$RLV IPW$RSV IPW$RSW
Disk / Tape Service	IPW$RDD IPW$RDQ IPW$RDT

Called by IPW$GDR	
Module	**Description**
IPW$$LW	Logical Writer
IPW$$NT	Network Transmitter
IPW$$OF	Offload Queues
IPW$$XJ	Execution JECL Scanner
IPW$$XR	Execution Reader

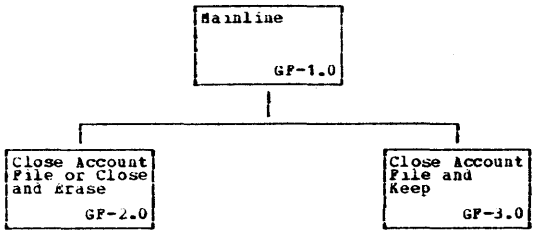
Labels	Chart GD: IPW$$GD - Get Data Record		Modified Data Fields	Reg. Usage	Calls
GDSD	The first 16 bytes constitute the section descriptor: 'GDCS release'				
GD00	Function Entry				
	Save caller registers 14 through 9 inclusive (IPW$SAV).		IPW$DSV		
	If a data buffer is present, branch to.....>	GD02			
	Reserve a new data buffer.				IPW$RSW
	Store buffer pointers.		TCDA(IPW$DTC) TCDV(IPW$DTC)		
	Force end of buffer.		TCPR(IPW$DTC)		
GD02	Check the buffer address. If the buffer is not empty, branch to.....>	GD40		R8	
	If the data file is located on disk, branch to.....>	GD10			
	Tape Spooling: Get First(Next) Data Block Record				
	Set the function track byte to C'G'.		TCFT(IPW$DTC)		
	Read tape data block. Since the tape might have been written with a different data block size, the first data block is read with max. data block size. The residual count is then used to calculate the actual data block size.				IPW$RDT
	If wrong-length record, insert stop code in TCB and return to caller....>	GDRT	TCTT(IPW$DTC)		
	Branch to continue.....>	GD40			
	Single Disk I/O Buffer: Get First(Next) Data Block Record				
GD10	If double buffer processing, branch to.....>	GD20			
	Check if the current data block still belongs to the track group or block group; link to.....>	GD70		R14	
	If yes, branch to.....>	GD15			
	Otherwise, get next queue record and first data block; link to.....>	GD60		R14	
	Branch to continue.....>	GD40			
GD15	Get next data block, branch to.....>	GD65		R14	
	Continue; branch to.....>	GD40			

Labels	Chart GD: IPW$$GD - Get Data Record		Modified Data Fields	Reg. Usage	Calls
	Get First(Next) Data BLock Record				
	Double Disk I/O BUffering: - Get First(Next) Data BLock Record				
GD20	Load register 14 with return address.			R14	
	If the second data block is available link to GD60 and branch to.....>	GD24			
	If this is the first time through, link to GD60 and branch to.....>	GD24			
	Swap the buffer pointers.		TCDW(IPW$DTC) TC2DW(IPW$DTC)	R0,R1	
	Indicate that the buffer is empty.		TCPR(IPW$DTC)	R8	
GD24	Address the disk request word by using Register 3.				
	Increment disk address (update the record address to the next record on the track), branch to.....>	GD80	TCDW(IPW$DTC)	R3,R14	
	Check if the next data block belongs to the same track group, branch to..>	GD70		R14	
	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)		
	Issue a read block to make sure that previous I/O is completed.				IPW$RDD
	Indicate next block is not read in, branch.....>	GD40	TC2DW(IPW$DTC)		
GD30	Set function track byte to C'G'. Read in next data block in the second data buffer.		TCFT(IPW$DTC)		IPW$RDD
	Get Extended Length Data Record				
GD40	If record is extended (>DLBL-8 bytes) then branch .....>	GX00	DRGP(IPW$DDR)		
	If the previous record was not extended, then branch .....>	GD43	TCGP(IPW$DTC)		
	Free the last extended record buffer.			R1	IPW$RLV
	Branch to .....>	GD43			
GX00	If it not the first extension, then branch .....>	GX08	DRG3(IPW$DDR)		

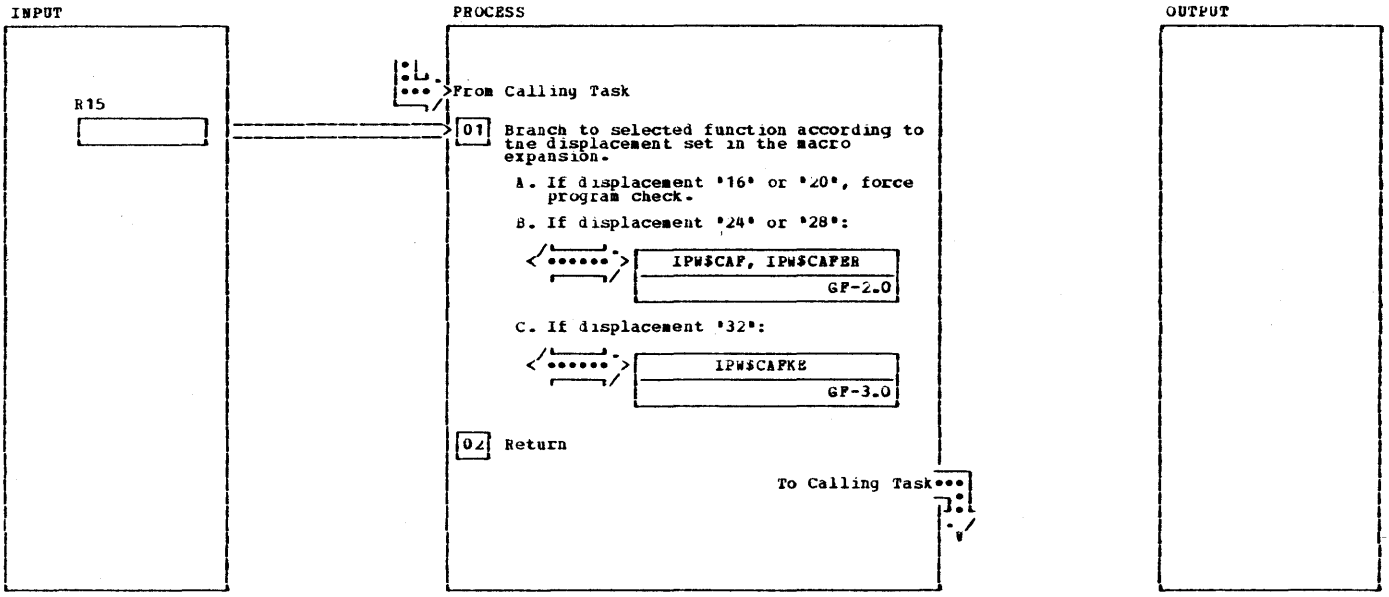
Labels	Chart GD: IPW$$GD - Get Data Record		Modified Data Fields	Reg. Usage	Calls
	Reserve a buffer for the present extended record space. If the buffer request was not successful, then return to the caller	GDRT	DREL(IPW$DDR)	R1	IPW$RSV
	Store buffer pointer and the extended record length the TCB.		TCRV(IPW$DTC) TCRL(IPW$DTC)	R2	
GX08	Move the record extension to the extended record buffer.			R6,R7 R8,R9	
	If this is the last extension of the extended record, branch	GX09	DRG3(IPW$DDR)		
	Indicate the data block is empty.		TCPR(IPW$DTC)		
	If tape spooling or double buffering then branch to read the next data block	GD02	TCSI(IPW$DTC) TCDB(IPW$DTC)		
	Link to increment the disk request word	GD80			
	Branch to read the next data block .>	GD02			
GX09	Reload register 8 pointer to block.			R8,R2	
	Update Caller's TCB				
GD43	Locate the next record within the block and set the appropriate values in the record control word:				
	• Command code. • General purpose byte (1). • General purpose byte (2) if internal control record (TCCC=x'ff') • Data Record pointer in Data Block.		TCCC(IPW$DTC) TCGP(IPW$DTC) TCG2(IPW$DTC) TCPR(IPW$DTC)		
GD46	If not an extended record, then: • Data Record address. • Data Record length.		TCRV(IPW$DTC) TCRL(IPW$DTC)		
GD48	Check for a break condition or end of block. If not, branch to.....	GDRT			

Labels	Chart GD: IPW$$GD - Get Data Record		Modified Data Fields	Reg. Usage	Calls
	Increment Disk Address:				
	Reset record pointer to start of block.		TCPR(IPW$DTC)		
	In case of tape spooling, exit.....>	GDRT			
	In case of double buffer processing, branch to.....>	GDRT			
	Update the record address to the next record on the track; branch to.....>	GD80	TCDW(IPW$DTC)	R1,R3 R14	
GDRT	Function Exit				
	Reset function track indicator to I.		TCFT(IPW$DTC)		
	Restore registers and return to caller.	(R14)		R14-R9	
	Get Next Track Group or Block Group:				
	This subroutine reads the next queue record in set and/or reads in the next data block respectively.				
GD60	Set function track byte to C'S'.		TCFT(IPW$DTC)		
	Save the job suffix number.			R4	
	Read in next queue record.				IPW$RDQ
	Restore the job suffix number.		QRSN(IPW$DQR)	R4	
	Update the disk address to the first data block of the new track group.		TC2DW(IPW$DTC)		
	If double buffering, set disk address in 2nd I/O request word.		TC2DW(IPW$DTC)		
GD65	Set the function track byte to C'G'.		TCFT(IPW$DTC)		
	Read in the data block.				IPW$RDD
	Return to caller.....>	(R14)			

Labels	Chart GD: IPW$$GD - Get Data Record		Modified Data Fields	Reg. Usage	Calls
	Subroutine to Check Requirement for a new Track/Block Group:				
GD70	Get address of first MCB.			R1	
	If data file not on FBM device.....>	GD74			
	If this block will fit into the block group, return to caller.....>	(R14)			
	If not, return to caller.....>	4(R14)			
GD74	If the new data block belongs to the same track, return to caller.....>	(R14)			
	If the next block is on the next track, update the disk address (track number).		IORG		
	If the next track belongs to the same track group, return to caller.....>	(R14)			
	Otherwise, return to caller via register 14 with a displacement of 4.	4(R14)			
	Subroutine to Increment Disk Request Word:				
GD80	Get address of first MCB.			R1	
	If data file not on FBM device.....>	GD82			
	Increase the block number by unit of transfer and store it.		IOBF		
	Return to caller.....>	(R14)			
GD82	Update the seek address of the data block: The record number is incremented by one.				
	Return to caller.....>	(R14)			



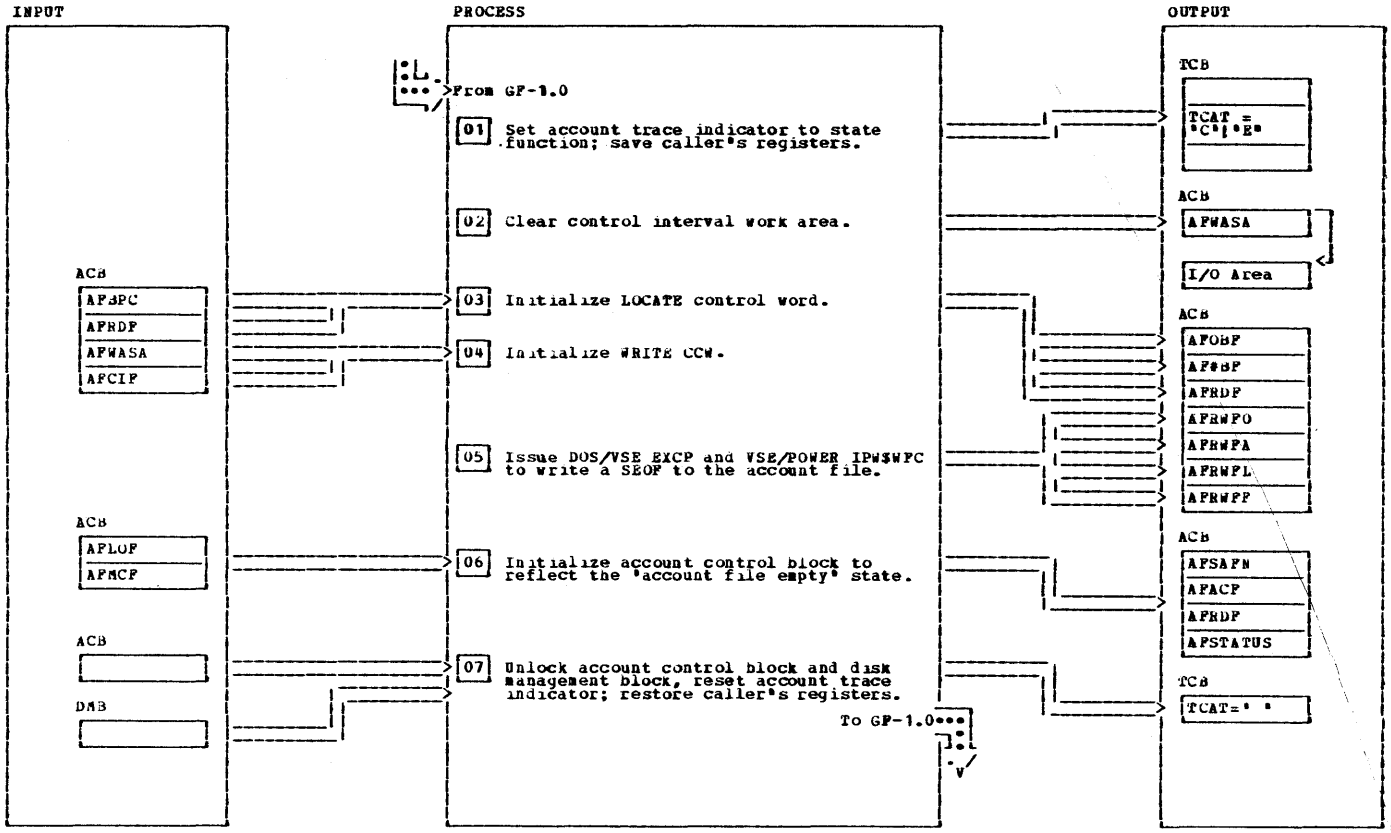
IPW$$GF - Mainline



NOTES	MODULE	LABEL	REF
The mainline consists of a branch table to two different routines. The macro expansions of the macros IPWOAF,IPWGAR,IPWCAF, IPWCAF ERASE, and IPW$CAF KEEP load the address of the proper branch instruction in the branch table, and pass control to it.		GPSTART	
1 A. The macros IPW$OAF and IPW$GAR are no longer required for VSE/POWER account files that reside on FBA devices and are not used. If the above-noted entry points are chosen, a program check is forced. B. The routine is entered at two entry points by macros IPW$CAF and IPW$CAF ERASE, which set the proper account trace indicator into the TCB and saves the caller's registers in the register save area that is addressed by register 13 before joining a common routine.			

NOTES	MODULE	LABEL	REF
C. The routine is entered at one entry point by macro IPW$CAF KEEP.			$SAV

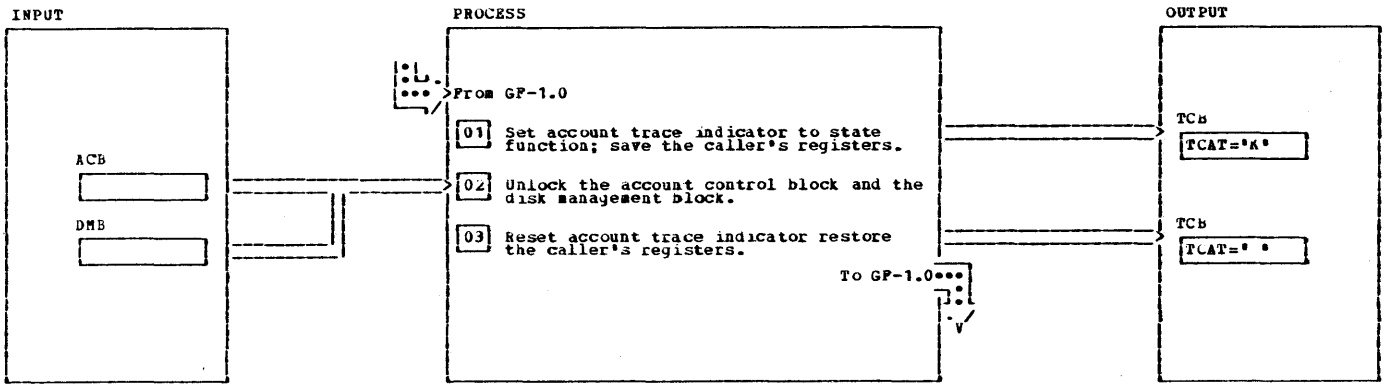
IPW$$GF - Close Account File, Or Close Account File And Erase



NOTES	MODULE	LABEL	REF
These functions initialize the put account work area to binary zeros, and write it to the account file to form a SEOF-CI (Software End of File Control Interval) on the first CI of the account file. Then the account control block is formatted to allow put account to start at the beginning of the account File (cold start format). The routine has two entry points to accommodate IPW$CAF and IPW$CAF ERASE macro requests.			
1 Two entry points are established for macros IPW$CAF and IPW$CAF ERASE to setup the proper account trace indicator. The caller's registers are saved in the register save area that is addressed by register 13. Then they join a common routine.		GP$CAF GP$CAFER	$SAV
2 A binary zero control interval is built to write a Software End of File record (SEOF) to the VSE/POWER account file.		GPCAF	

NOTES	MODULE	LABEL	REF
3 The LOCATE control word and CCWs, located in the account control block, are initialized to point to the very first control interval on the VSE/POWER account file, and to write the SEOF.			
5 Write to disk.	EXCP		$WPC
6 The account control block is set to an initial value to represent the empty account file state. This allows put account to start at the lower limit of the file with the next write request and to continue if a 'wait on full file' state was encountered.			
7 This is done to allow next put account and/or save account to control the account file. The caller's registers are restored from the register save area that is addressed by register 13.			$RLR $RET

IPW$$GF - Close Account File And Keep

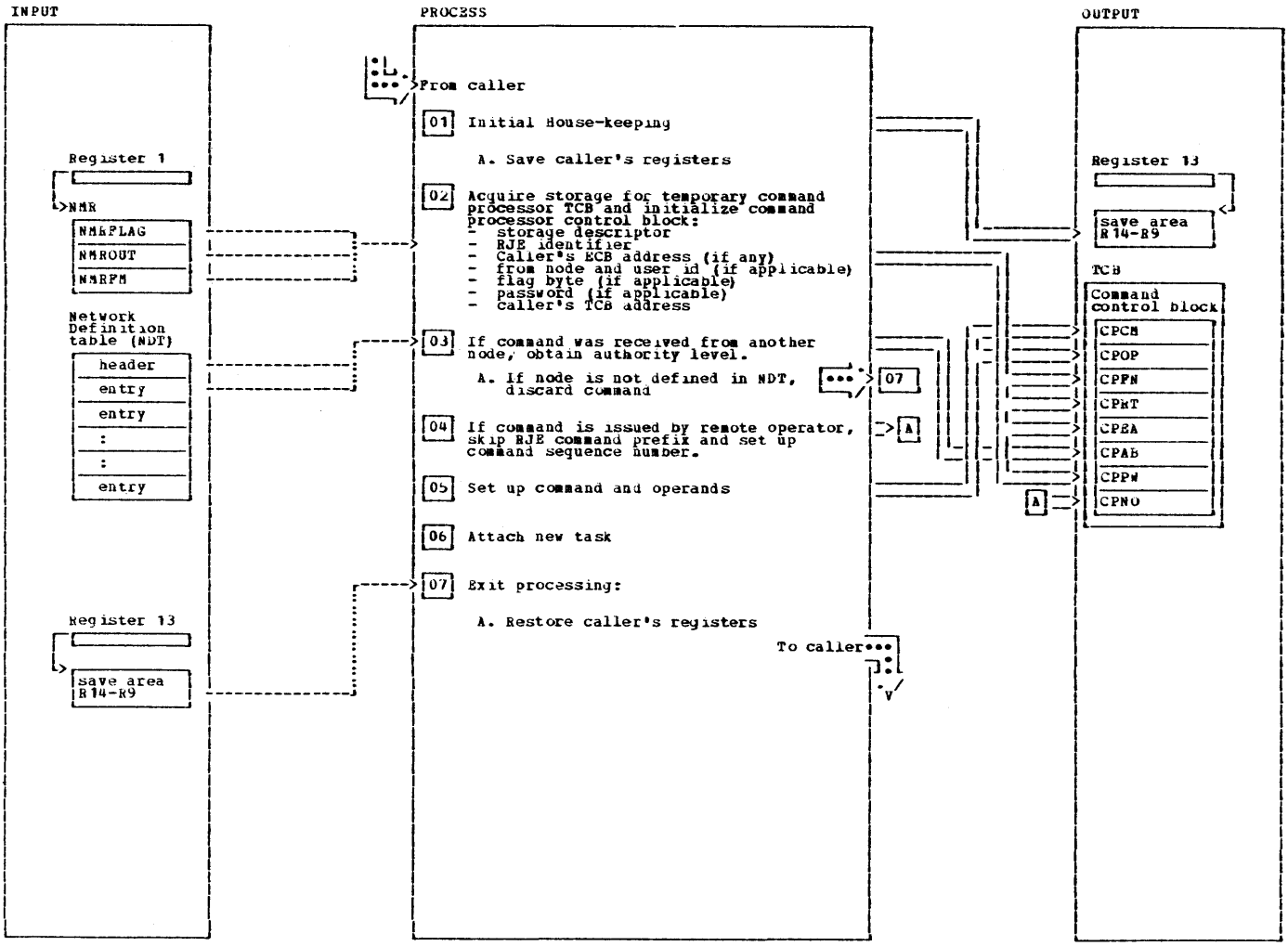


NOTES	MODULE	LABEL	REF
1 The caller's registers are saved in the save area that is addressed by register 13.			$SAV

NOTES	MODULE	LABEL	REF
2 The account control block and disk management block are unlocked.		GF$CAFKE	$RLR
3 Restore the caller's registers from the save area that is addressed by register 13.			$RET

CHART IC: IPW$$IC

IPW$$IC - INVOKE COMMAND PROCESSOR

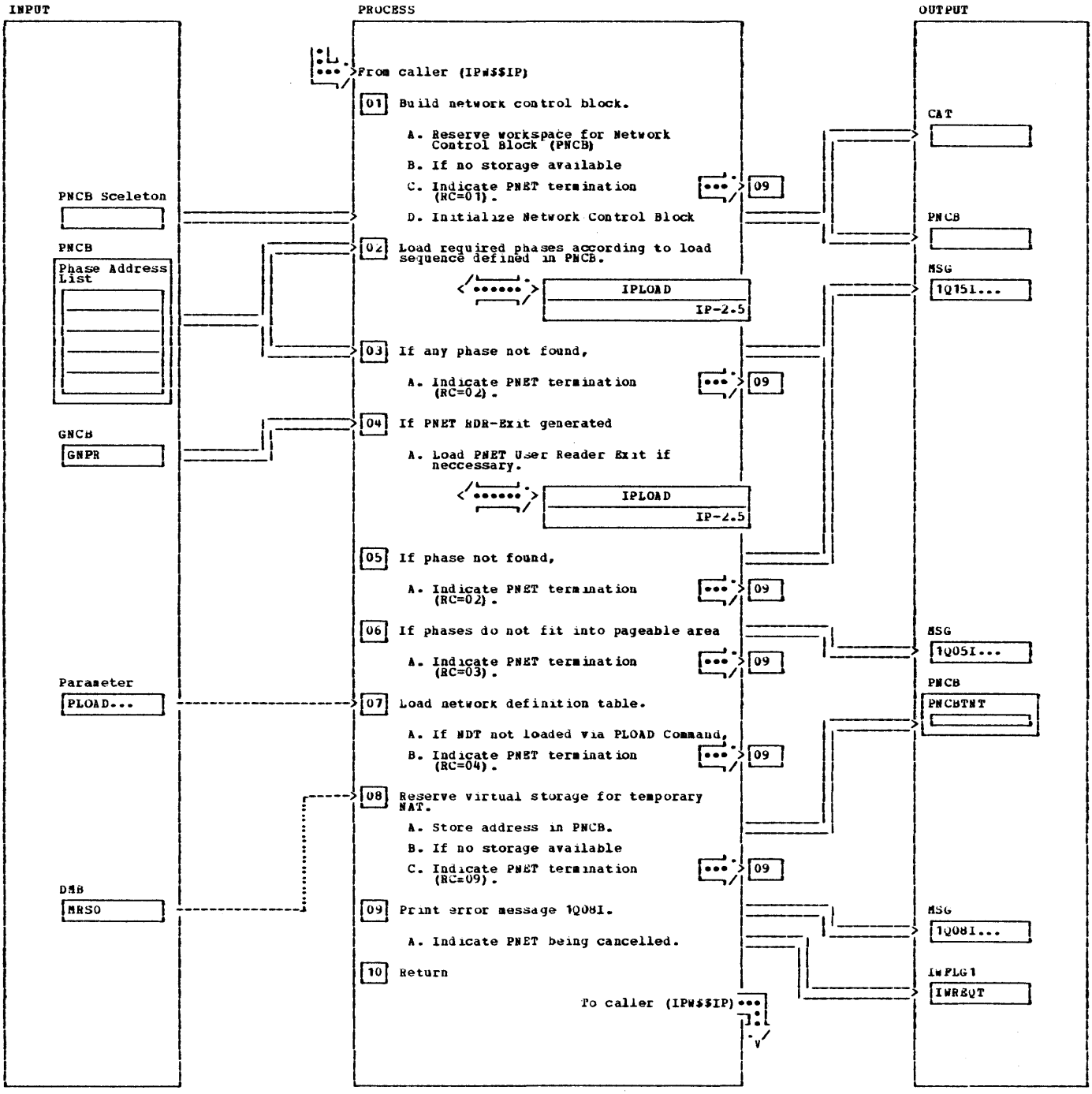


NOTES	MODULE	LABEL	REF
1 The caller's registers are saved in the save area addressed by register 13.			$SAV
2 Storage is reserved for the new command processor task control block. The command control block which is part of the TCB is formatted. For a CTLSPOOL request , the password is taken from the SPL parameter list.		IC00	$RSW
3 The network definition table (NDT) is scanned to locate the originator's node entry. The assigned authority level is then moved to the command control block. If the command was issued from a remote or interactive user, the authorization level is degraded up to a maximum of 'JOB' authority.			
If however if the node is unknown, the just acquired TCB storage is released and the command is discarded.			$RLW

NOTES	MODULE	LABEL	REF
4 The command passed by the caller is examined for the RJE prefix (* ..). If this is found, the sequence number is saved in the command control block.		IC12	
5 The command is scanned to locate the operation code (command verb). If the first character of the command is 'P', the character is stripped off, unless the command is specified is its short form. Next the command is scanned to locate the start of the operands. The operands are then moved to the command control block.			
6 The address of the spool management parameter list (SPL) is initialized, if present.		IC50	$ATT
7 The caller's registers are restored from the save area addressed by register 13 and return is made to the caller			$RET

This page was left blank intentionally.

IPW$$IN -PNET FEATURE INITIALIZATION

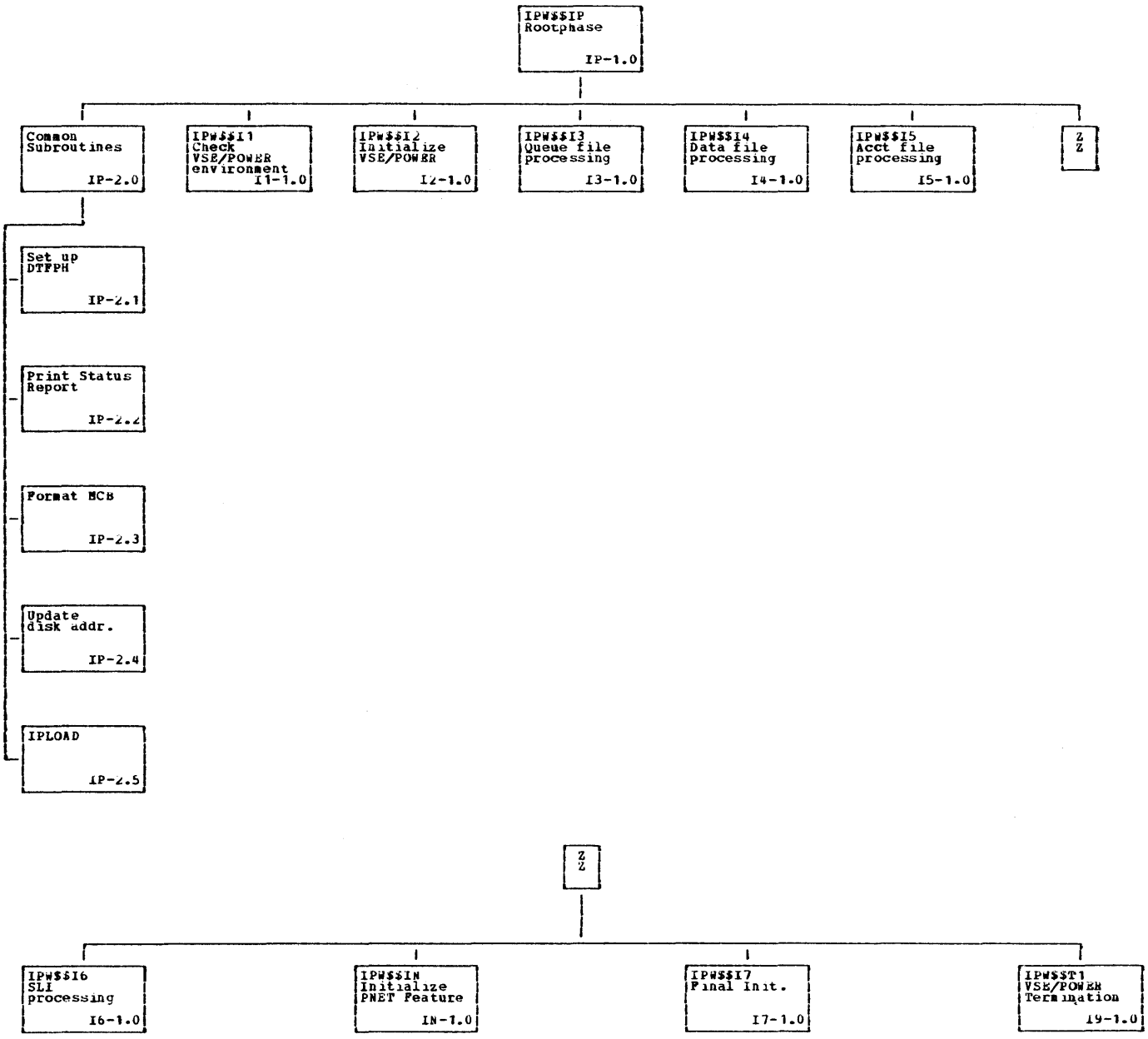


IPW$$IN -PNET FEATURE INITIALIZATION

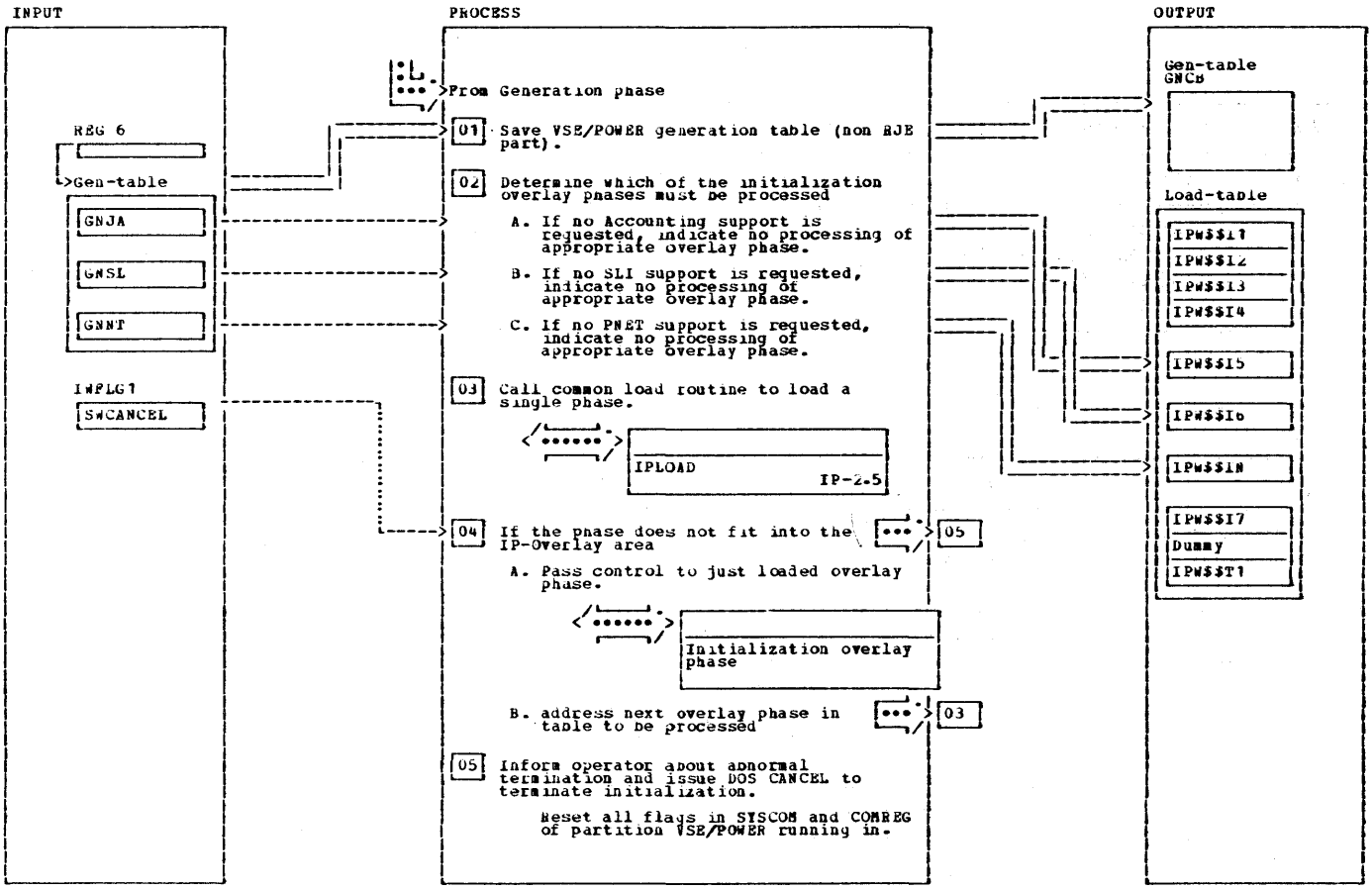
NOTES	MODULE	LABEL	REF
1 Real storage is reserved in the length of the Network Control Block (PNCB). If no storage is get, Reason Code 1 is set and PNET Initialization is cancelled. Otherwise the address field of the CAT is updated to anchor the PNCB to the CAT of VSE/POWER. C. After storage has been reserved, the Skeleton of the PNCB is than copied into the reserved storage.		CACP	$RSW
2 The IPW$$IP Subroutine is called to load the required PNET phases into the Pageable area of VSE/POWER. On return, the IPLOAD has loaded the phases into the Pageable Area of VSE/POWER.		IPLOAD	$IP
3 If any phase was not found, IPLOAD has set IWCANCEL ,Reason Code 2 is set to cancel PNET Initialization.			
4 The generation table is cnecked to see whether an user PNET RDR-Exit is specified or not. If specified, the phasename from the generation table is used to load the PNET User Reader Exit.			
5 If there is no reader exit available, phase not found Reason Code 2 is set to terminate PNET Initialization.			

NOTES	MODULE	LABEL	REF
6 The highest virtual partition address is compared against the end address of the last phase to see if the phases fit into the pageable area of VSE/POWER. If not, Reason Code 3 is set to terminate PNET initialization.			
7 The phasename of the Network Definition Table from the GENTAB is used to create a PLOAD NET=phasename command and a temporary command processor is invoked to load the Network Definition Table (NDT). On return from command processor, the CP has stored the address of tne NDT into the PNCB if the load was successfully, otherwise the address field contains zeros. In this case, Reason Code 4 is set to terminate PNET.			$ICP
8 If shared spooling is supported virtual storage is reserved for the temporary Node Attach Table. If no storage available rc 09 is set and initialisation of pnet is terminated.			$RSV
9 PNET could not be initialized and therefor the Message: 1008I UNABLE TO INITIALIZE PNET FEATURE RC=nnn is printed and the cancel request IWREQT is set.		IWFLG1	$GAM

CHART IP: IPW$$IP



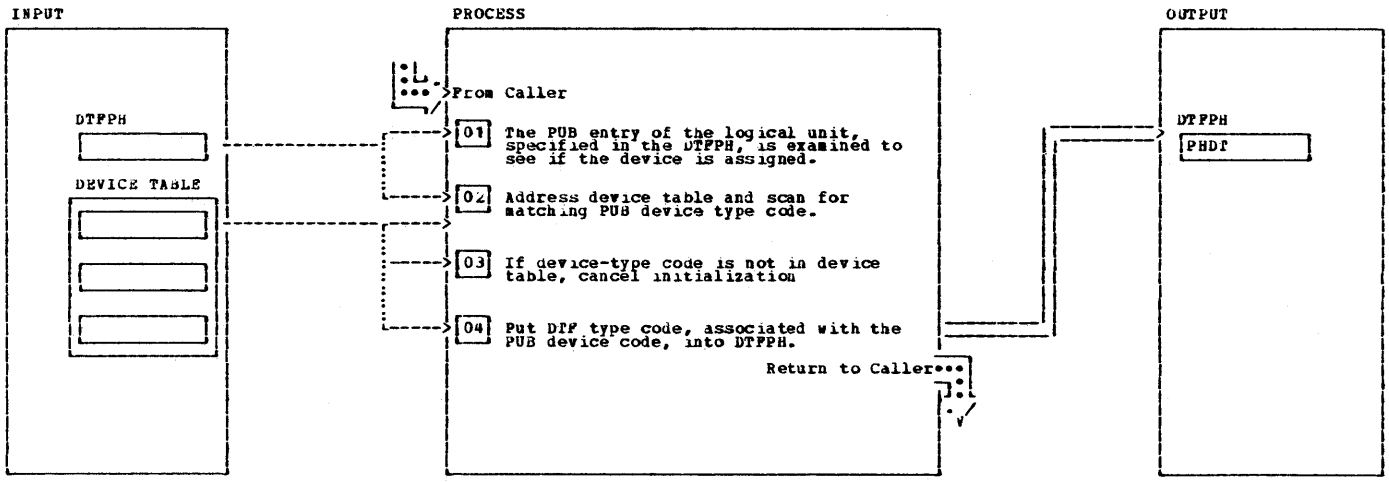
IPW$$1P - Initialization Processor (Rootphase)



NOTES	MODULE	LABEL	REF
The loader in front of the generation table loads the initialization rootphase behind the first page in the pageable area. (The first page is reserved for the permanent command processor as work area).			
1 The non-RJE part of the VSE/POWER generation table is saved within the IP rootphase.			

NOTES	MODULE	LABEL	REF
3 The rootphase is responsible to check the length of the phase to be loaded before loading. Control is passed to the IPLOAD Subroutine to get the length of the phase.		IPLOAD	
4 This process keeps going until the last overlay phase 'IPW$$T1' is loaded and has received control.			

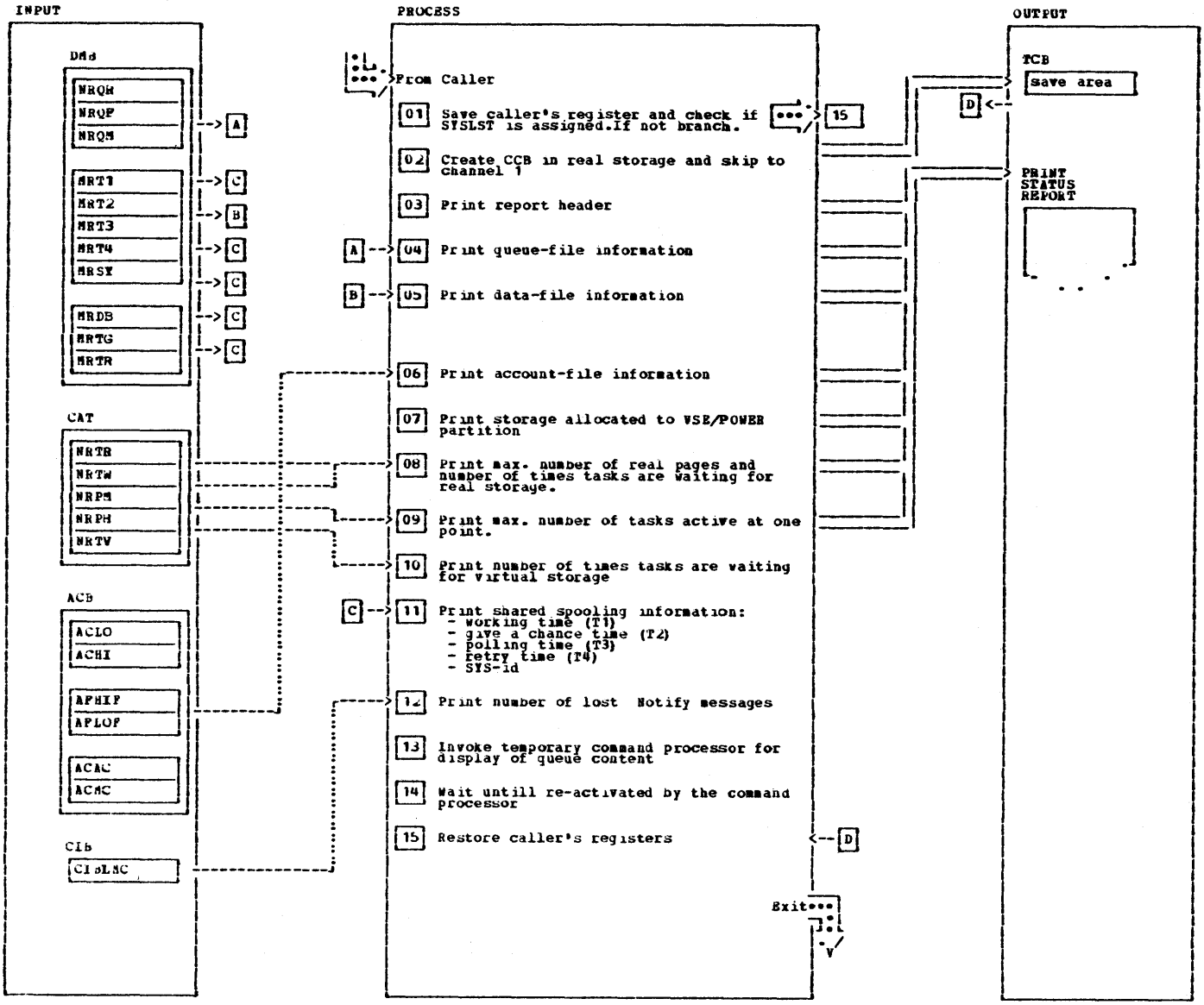
Subroutine: Set up DTFPH routine



NOTES	MODULE	LABEL	REF
**********			
The PUB device type code is obtained via the logical unit specified in the DTFPH. This code is checked against the supported device-type codes in the device table. If no match is found, message 1Q07I is issued and the initialization is terminated. If a match is found, the proper device-type code is inserted in the DTFPH. The address of the device characteristics is returned in register 2 and the default datablock size is returned in register 1.			
**********			

	NOTES	MODULE	LABEL	REF
1	The GETVCE macro is issued to obtain the device characteristics of the disk device containing the file assigned to the programmer logical unit specified in the DTFPH. If a bad (non zero) returncode is given back, message 1Q07I INVALID LOGICAL UNIT 'XXXXXXXX' is issued and the initialization in canceled.	GETVCE	SD00	
2	The internal device table, which contains an entry for each disk device supported by VSE/POWER, is scanned to locate the entry for which the PUB device type matches.			$CNC
3	If device-type code is not in device table, print message 1Q07I INVALID LOGICAL UNIT and cancel initialization.			$GAM
4	Insert DTF device-type code in DTFPH.		SD16	$CNC

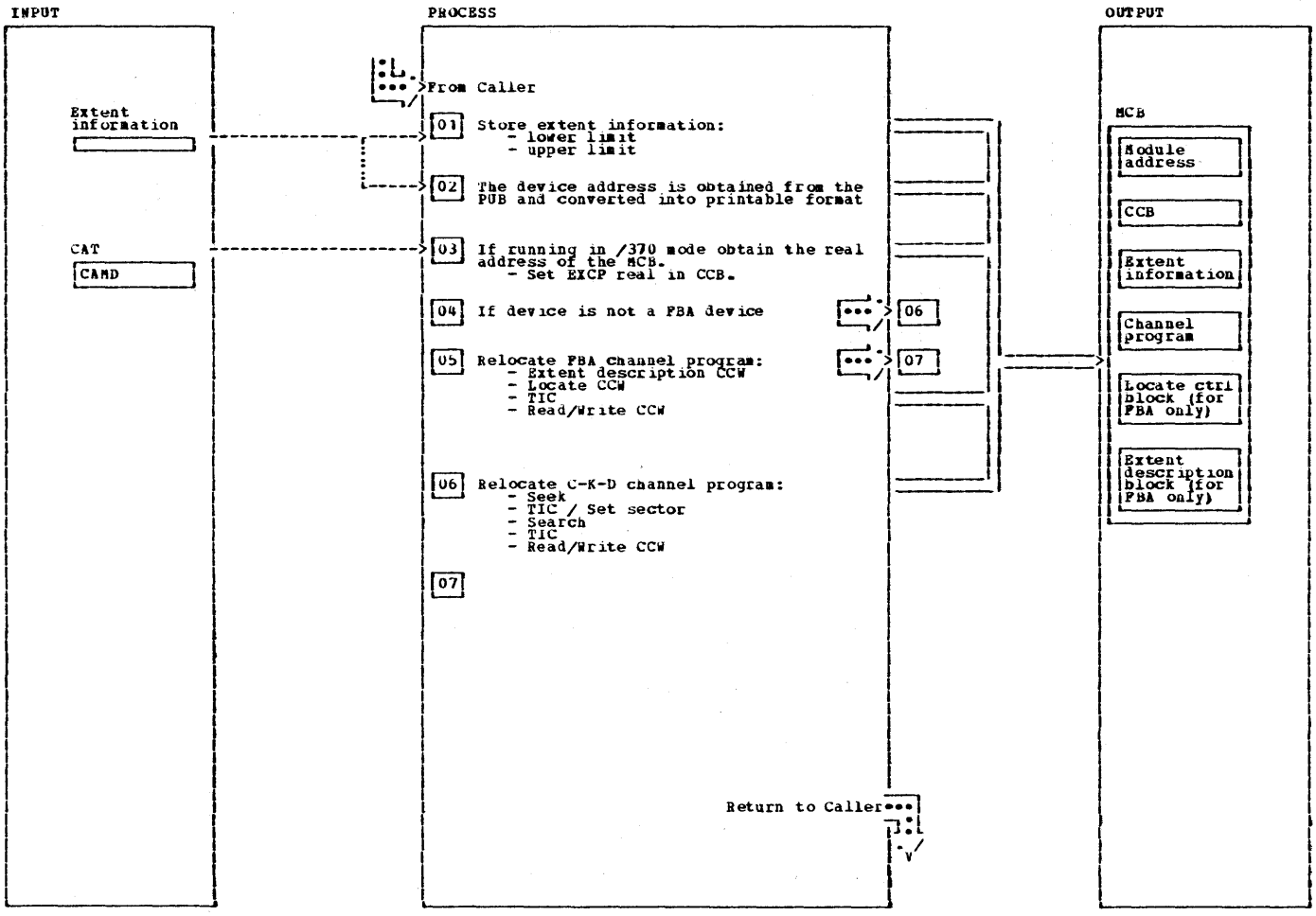
Subroutine: Print Status Report



	NOTES	MODULE	LABEL	REF
1	The DOS/VSE internal macro SYSIR is used to obtain LUB and PUB address to find out if SYSLST is assigned or not. A Status Report is provided if SYSLST is assigned to a printer device	SYSIR	PS00	$SAV
2	If the printer is a 3800, an initialize printer is done instead of the skip to channel one	$RSW		
3				
4	Queue file information: - Header - number of queue records (NRQR) - number of free q-records (NRQF) - number of q-records used (NRQM)	COMRG	PS10	$RDC

	NOTES	MODULE	LABEL	REF
5	Data file information: - Header - number of tracks/FBA blocks for Data file - trackgroup/blockgroup size C-K-D - Data blocksize (NRDB)			
6	Account file information: - Header - number of tracks (ACLO/ACHI) - number of blocks (AFHIF/AFLOF) for FBA device - precentage of account file (AFHIF/AFLOF)		PS50	
13	The command processor is invoked to display the content of all queues	EXCP		$ICP
13 15				$WFC $RET

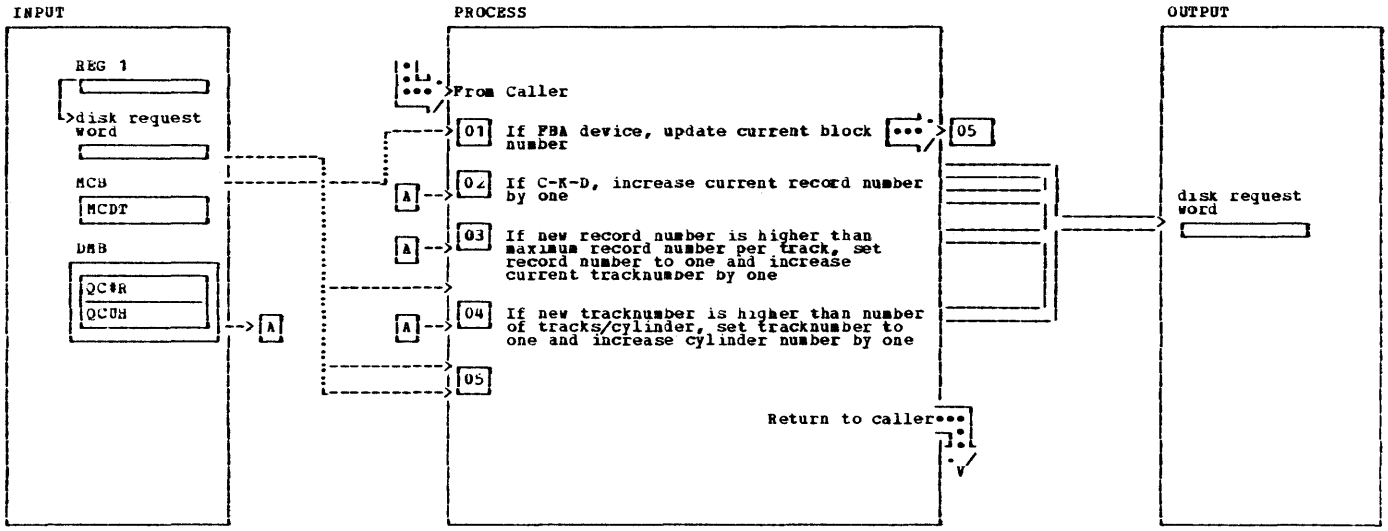
Subroutine: Format MCB for data file and queue file



NOTES	MODULE	LABEL	REF
1 The extent information obtained from the XTNTXIT processing is saved in the module control block (MCB)	EXTRACT	FM00	
2 Get the physical device address out of the PUB and convert it to a printable format. Save device address in the storage descriptor of the MCB.			

NOTES	MODULE	LABEL	REF
3 The real address of the MCB, which will be used later on for relocation of the channel program, is obtained when running in /370 mode	REALAD		

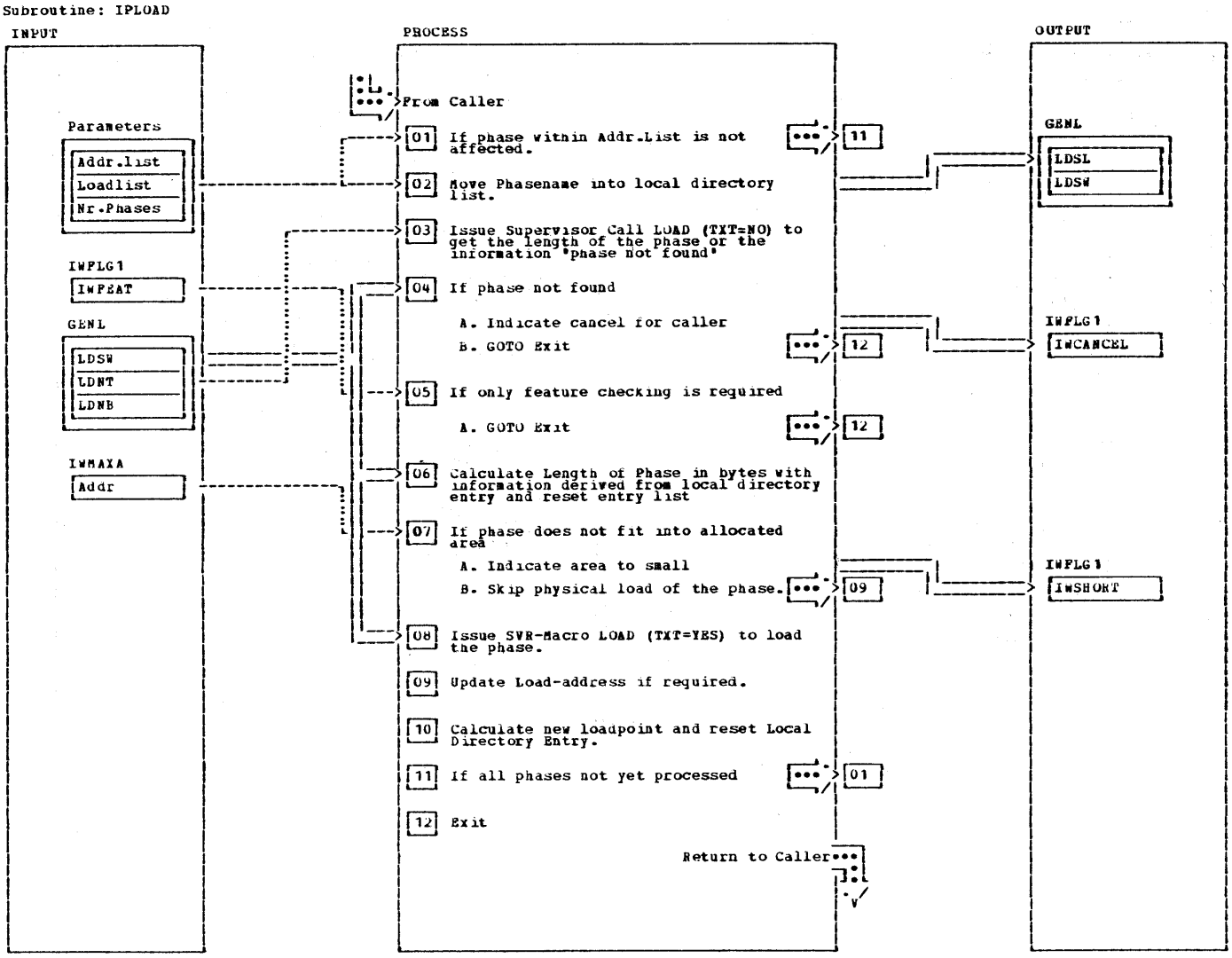
Subroutine: Update disk adress



NOTES	MODULE	LABEL	REF
This subroutine updates the disk address, pointed to by register 1, to address the next queue record.		UPDT	

NOTES	MODULE	LABEL	REF
1 Increase the current blocknumber by one and store back result			

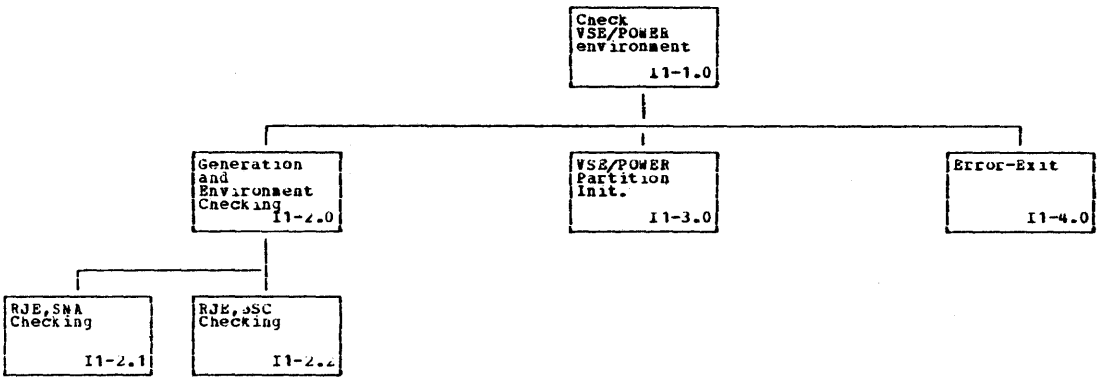
This page was left blank intentionally.


Subroutine: IPLOAD
INPUT
PROCESS
OUTPUT
Parameters
Addr.list
Loadlist
Nr.Phases
IWFLG1
IWFEAT
GENL
LDSW
LDNT
LDNB
IWMAXA
Addr
From Caller
01 If phase within Addr.List is not affected.
02 Move Phasename into local directory list.
03 Issue Supervisor Call LOAD (TXT=NO) to get the length of the phase or the information 'phase not found'
04 If phase not found
A. Indicate cancel for caller
B. GOTO Exit
05 If only feature checking is required
A. GOTO Exit
06 Calculate Length of Phase in bytes with information derived from local directory entry and reset entry list
07 If phase does not fit into allocated area
A. Indicate area to small
B. Skip physical load of the phase.
08 Issue SVC-Macro LOAD (TXT=YES) to load the phase.
09 Update Load-address if required.
10 Calculate new loadpoint and reset Local Directory Entry.
11 If all phases not yet processed
12 Exit
Return to Caller
GENL
LDSL
LDSW
IWFLG1
IWCANCEL
IWFLG1
IWSHORT

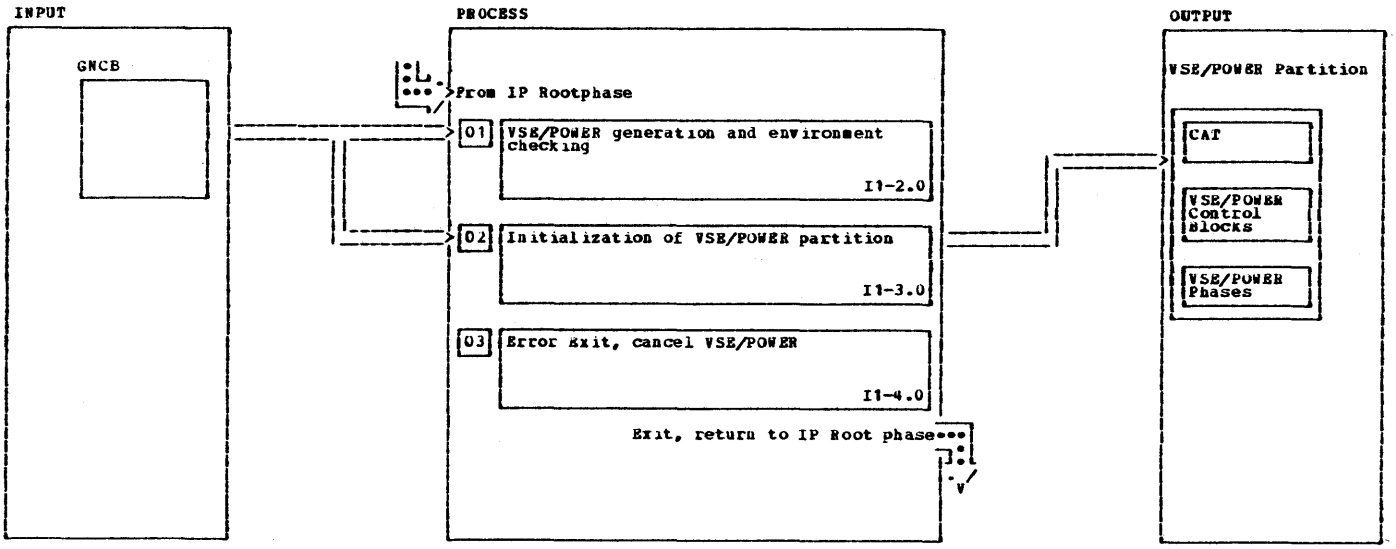
Subroutine: IPLOAD

NOTES	MODULE	LABEL	REF
**********		IPLOAD	
This subroutine is called under the following circumstances: 1. The length of a phase is required of 2. the phase has to be loaded from the CIL. **********			
1 The load sequence already defined within several control blocks (e.g. CAT, PNCB...) is addressed in REG.3. If a single phase is affected, REG.3 points to the full phasename elsewhere in the storage. The caller has loaded the number of phases in REG.4 and REG.2 points to the load-address, where it is to be loaded Every entry in the loadlist contains the following info: Byte 0+1 = phasename id (last two char. of phase) zero, phase not affected. byte 2 = blank byte 3 = A phase is to load at page boundary.			
2 Depending on the load to be performed, either a single phase or multiple phases is to be loaded, the Local Directory Entry Parameter is updated. Single phase = Move full phasename Multi. phases= Move POWER standard (IP##SS) and phasename Id from loadlist. If 'A' is indicated in the loadlist entry, the address where the phase is to be loaded is rounded up to the next higher page boundary.			

NOTES	MODULE	LABEL	REF
3 The SVR LOAD uses the Local Directory Entry to execute the required function.	LOAD		
4 If a phase is not found during length checking, indicated through x'04' in the Local Directory Entry the SWCANCEL is set and further processsing is stopped.			
5 If feature checking only is indicated,the physical load is skipped.			
6 The length of the phase is derived from the Local Directory Entry.			
7 The calculated length plus the load-address are compared against the end address of the area into which the phase has to be loaded. If the added value is greater than the end, no physical load is performed and error is indicated.			
8 The SVR now loads the required phase into the storage pointed to by REG-1.			
9 The Load-address in the affected Control Block (e.g. CAT....) is updated to anchor the phase to the common address list.			
10 The length of the phase is added to the phase begin address to get the new loadpoint.			
11 The number of phases to be processed, contained in REG.4 is decreased by 1 and REG.3 is set to the next entry of the loadlist. This keeps repeat of step 1 to 7 until REG.4 is zero.			



IPW$$I1 - Mainline

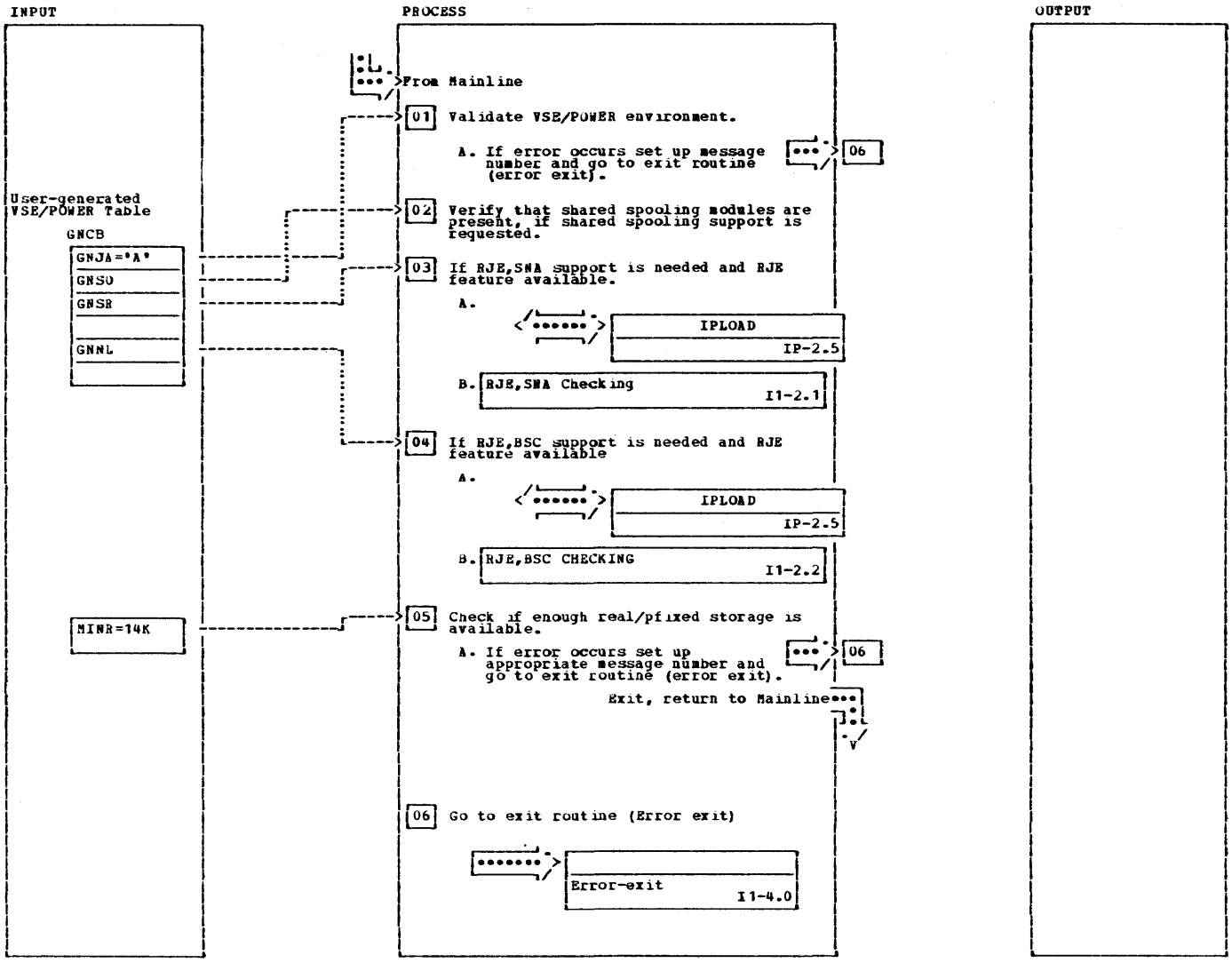


NOTES	MODULE	LABEL	REF
IPW$$I1 - Mainline checks the environment in which VSE/POWER has to be executed, loads all VSE/POWER phases, initializes the control address table (CAT) and RJE control blocks.			

NOTES	MODULE	LABEL	REF

This page was left blank intentionally.

IPW$$I1 - Generation and Environment Checking

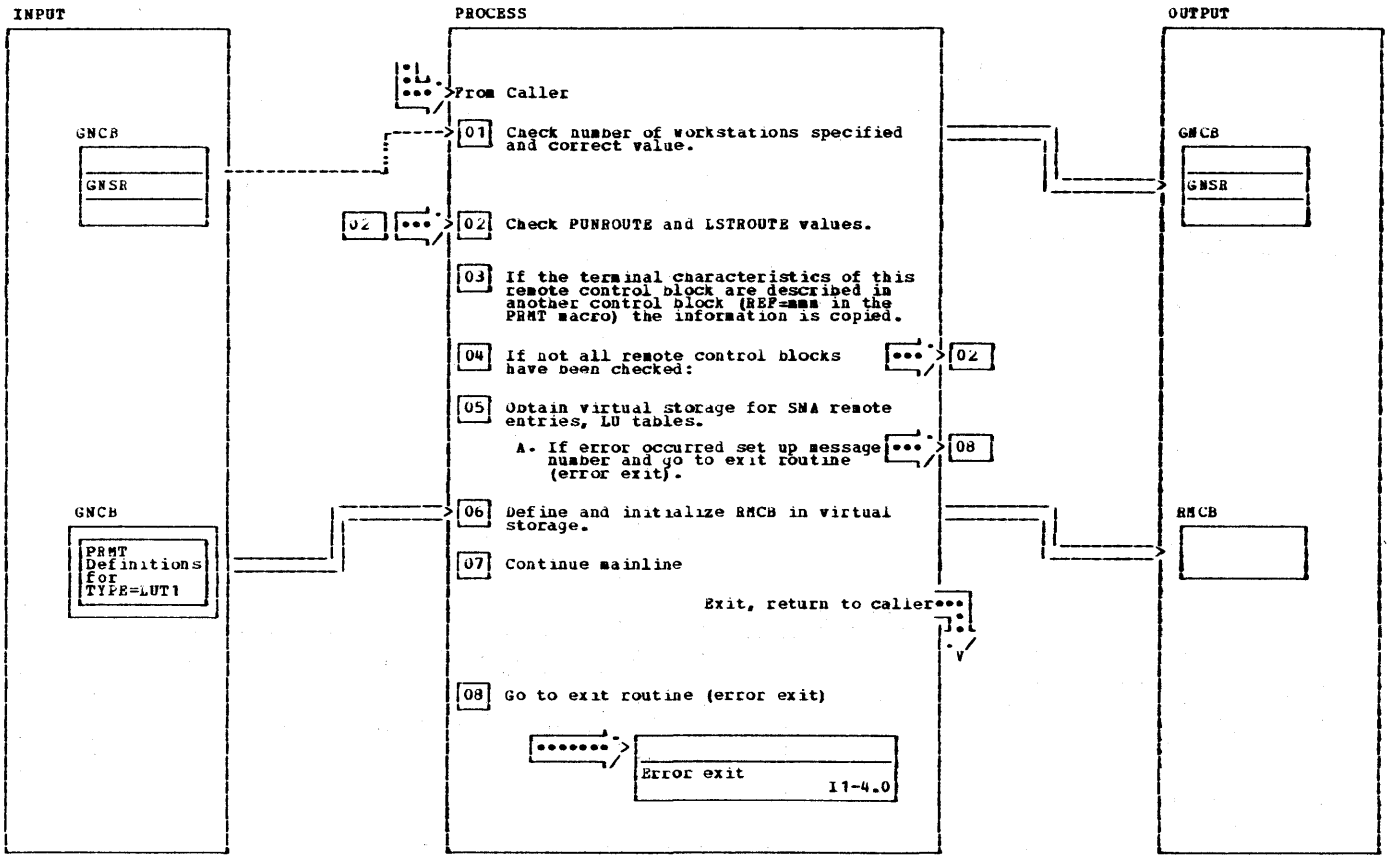


IPW$$I1 - Generation and Environment Checking

NOTES	MODULE	LABEL	REF
1 The following checks will be made:			
- if VSE/POWER is already active:	ASYSCOM	I100	
'1Q22I VSE/POWER ALREADY ACTIVE'	COMRG		
- if SYSLOG assignment not a 1052 CRT (model 115/125 integrated display operator console with 5213 console printer attached) or 3277:	SYSIB	I103	
'1Q06I SYSLOG NOT ASSIGNED TO CONSOLE'			
- if VSE/POWER is not a maintask:	RUNMODE		
'1Q02I VSE/POWER CANNOT RUN AS SUBTASK'			
- if VSE/POWER is not running in virtual mode:			
'1Q01I VSE/POWER CANNOT RUN IN REAL MODE'			
- if VSE/POWER generated with job accounting but JAI not present in supervisor:		I120	
'1Q10I SUPERVISOR WITHOUT ACCOUNTING SUPPORT'			
2 A dummy load will be issued to find out whether the TIMER module (IPW$$TI) is in the system. If the phase is not in the system, the feature is not supported and initialization is terminated. Message 1Q0AI SHARED SPOOLING FEATURE NOT SUPPORTED is issued.			

NOTES	MODULE	LABEL	REF
3 A dummy load will be issued to find out whether the SNA manager (IPW$$SN) is in the system. If the phase is not in the system, the feature is not supported and initialization is terminated. Message 1Q0AI RJE,SNA FEATURE NOT SUPPORTED is issued.		SR00	
4 Initiator will do a dummy load for BSC monitor (IPW$$BM). If the phase is not in the system, the feature is not supported and initialization is terminated. Message 1Q0AI RJE,BSC FEATURE NOT SUPPORTED is issued.		I1RJ	
5 If real/pfixed storage smaller than 14K display message: '1Q03I INSUFFICIENT REAL/PFIXED STORAGE ALLOCATED'.		1A10	
If the virtual pageable area is not large enough to get the initialization processor loaded on the right place, the VSE/POWER loader loads the processor behind the generation table. This causes that message 1Q05I PAGEABLE AREA TOO SMALL is issued.			

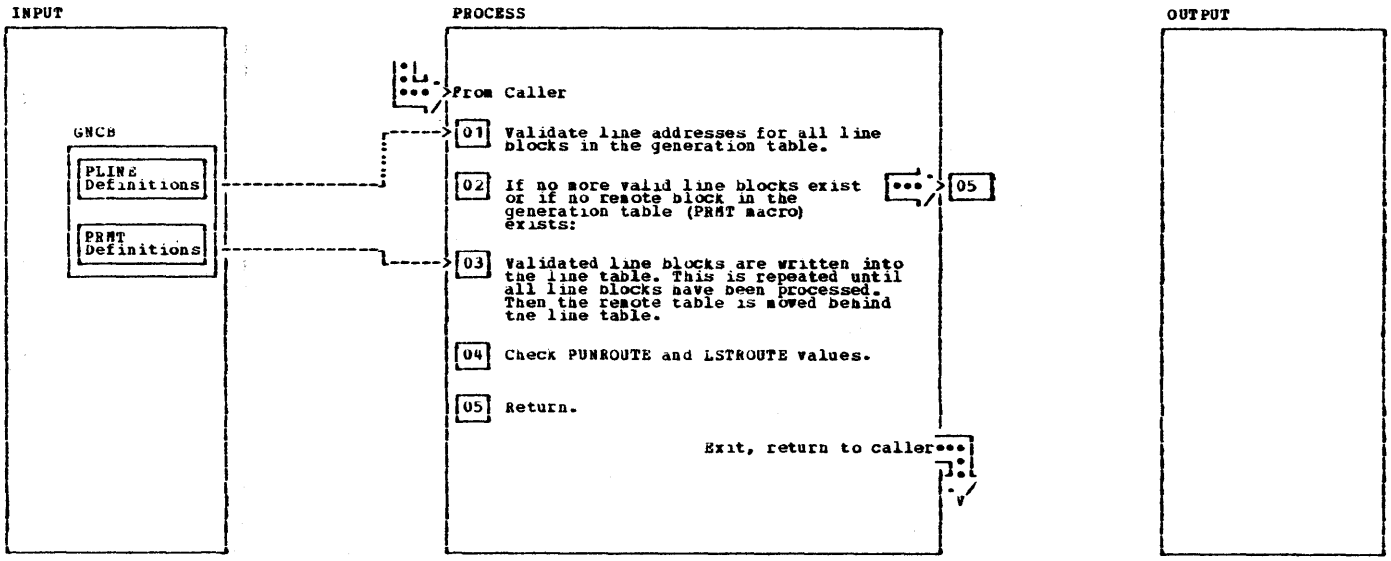
IPW$$I1 - RJE,SNA Checking



NOTES	MODULE	LABEL	REF
1 If no number of workstations is specified (macro POWER, parameter SNA=(0),=wscount) or if the number of workstations specified is higher than the number of SNA remotes (wscount > number of remote ID's), the number of workstations is set equal to the number of SNA remotes.		RS00	
2 The punch and list routing IDs are checked for validity, and if they are invalid, message: '1Q16I INVALID PUN (or LST) ROUTING FOR remid' is printed and the routing IDs are reset to X'00'.		RS05	
5 '1Q26I GETVIS AREA TOO SMALL'	GETVIS		

NOTES	MODULE	LABEL	REF
6 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. The SNA remote control block is initialized with the following fields from the VSE/POWER 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) 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.		RS47	

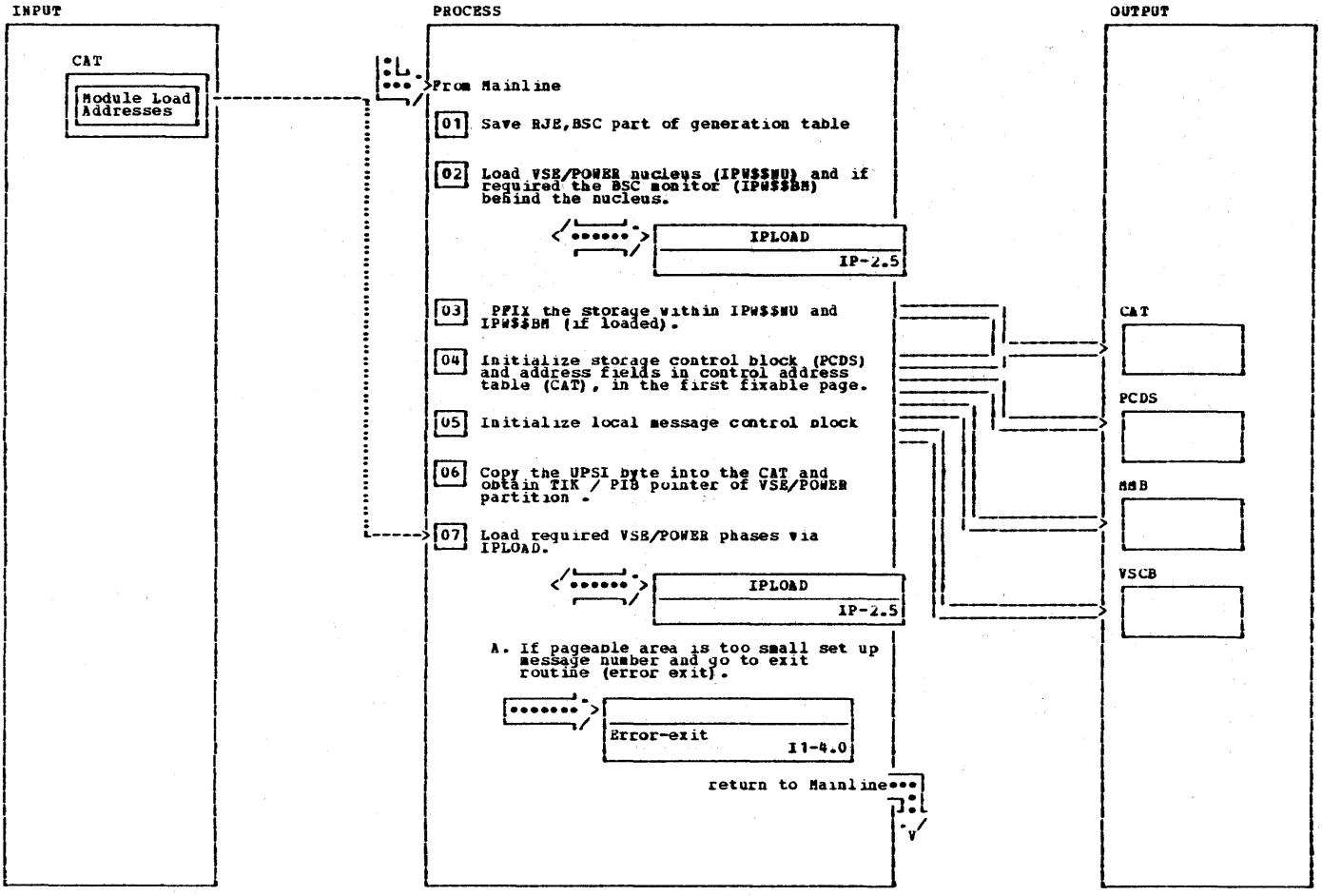
IPW$$I1 - RJE,BSC Checking



NOTES	MODULE	LABEL	REF
1 Check if the line blocks in the generation table have corresponding line addresses in the PUB. 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.		I1RJ	

NOTES	MODULE	LABEL	REF
4 The punch and list routing IDs are checked for validity and if they are invalid, message: '1Q16I INVALID PUN (or LST) ROUTING FOR remid' is printed and the routing IDs are defaulted to central punch/list routing (X'00').		RJ35	

IPW$$I1 - Initialization of VSE/POWER Partition

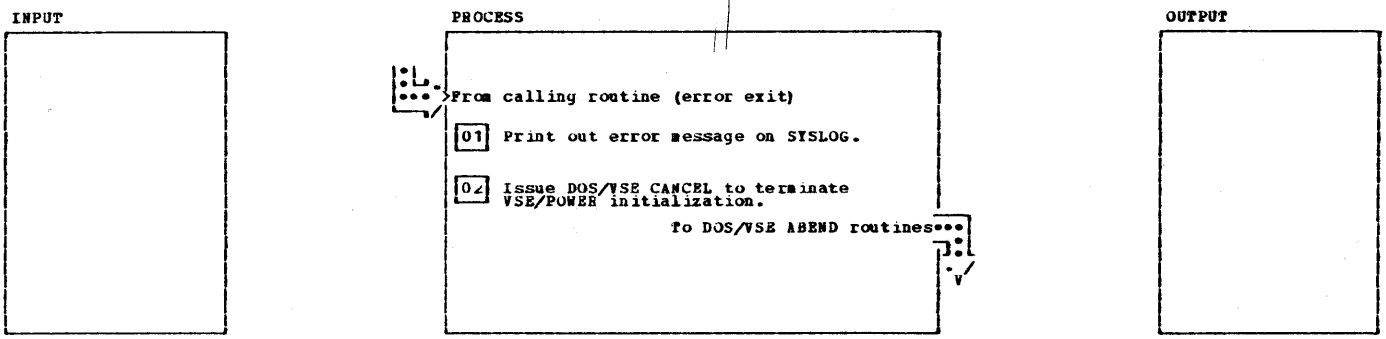


IPW$$I1 - Initialization of VSE/POWER Partition

	NOTES	MODULE	LABEL	REF
	The RJE,BSC part is saved in the next page behind IPW$$IP by generating two tables (LINE table and REMOTE table). The RJE,SNA part was already saved in GETVIS storage (control block RMCB).			
2	If RJE,BSC is supported, IPW$$BM will be loaded behind the DMB.			$IP
3	Behind the DMB or, if RJE,BSC is supported, behind IPW$$BM, the first fixable page of VSE/POWER fixable storage starts. This page contains initiator/terminator TCB, command processor TCB, WCB's for queue and data files, and free space.	PFIX	IA23	
4	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').	REALAD	IA27	
5	The local message control block is initialized by relocating the SYSLOG channel program: - read CCW - write CCW			
6	With the UPSI byte information VSE/POWER debug traps were activated.			
7	The phases of VSE/POWER BASE and the phases which apply to generation features are loaded and the phase lengths are summarized. When the phases do not fit into the pageable area, the following message is displayed: '1005I PAGEABLE AREA xx K TOO SMALL' Initialization is then terminated. The fetch sequence is defined in CAT (section '90.3 - module-load-addresses). For every phase a fullword is defined which will contain the load address. It contains the following information:			$IP

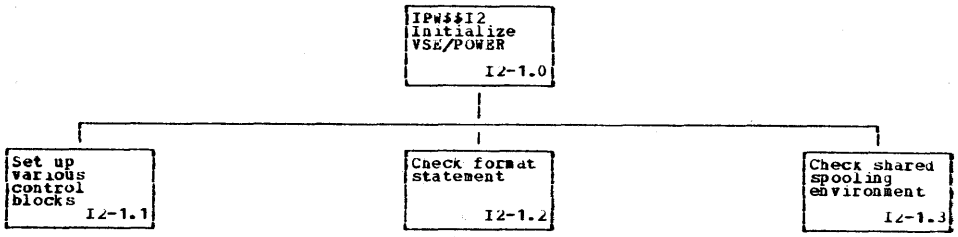
NOTES	MODULE	LABEL	REF
Byte 0 and 1: Last two characters of phase name (e.g. CM from IPW$$CM) = phase name id. Byte 2: Blank Byte 3: Page alignment indicator. 'A' means phase is loaded at page boundary ' ' means phase load address will not align to page boundary.			
If a phase is not needed, the appropriate module-load-address fields of the related phases in the CAT are set to hex zero (subroutine I1ZE) to suppress loading. If accounting is supported, the device type of the account file is checked whether it is FBA or CKD. If FBA, the IDs for the FBA phases will be moved into the module load address field in the CAT. The size of the BSC trace area, if required (defined in macro POWER), is also added to the total size of pageable area.	GETVCE	I138	

IPW$$I1 - Exit

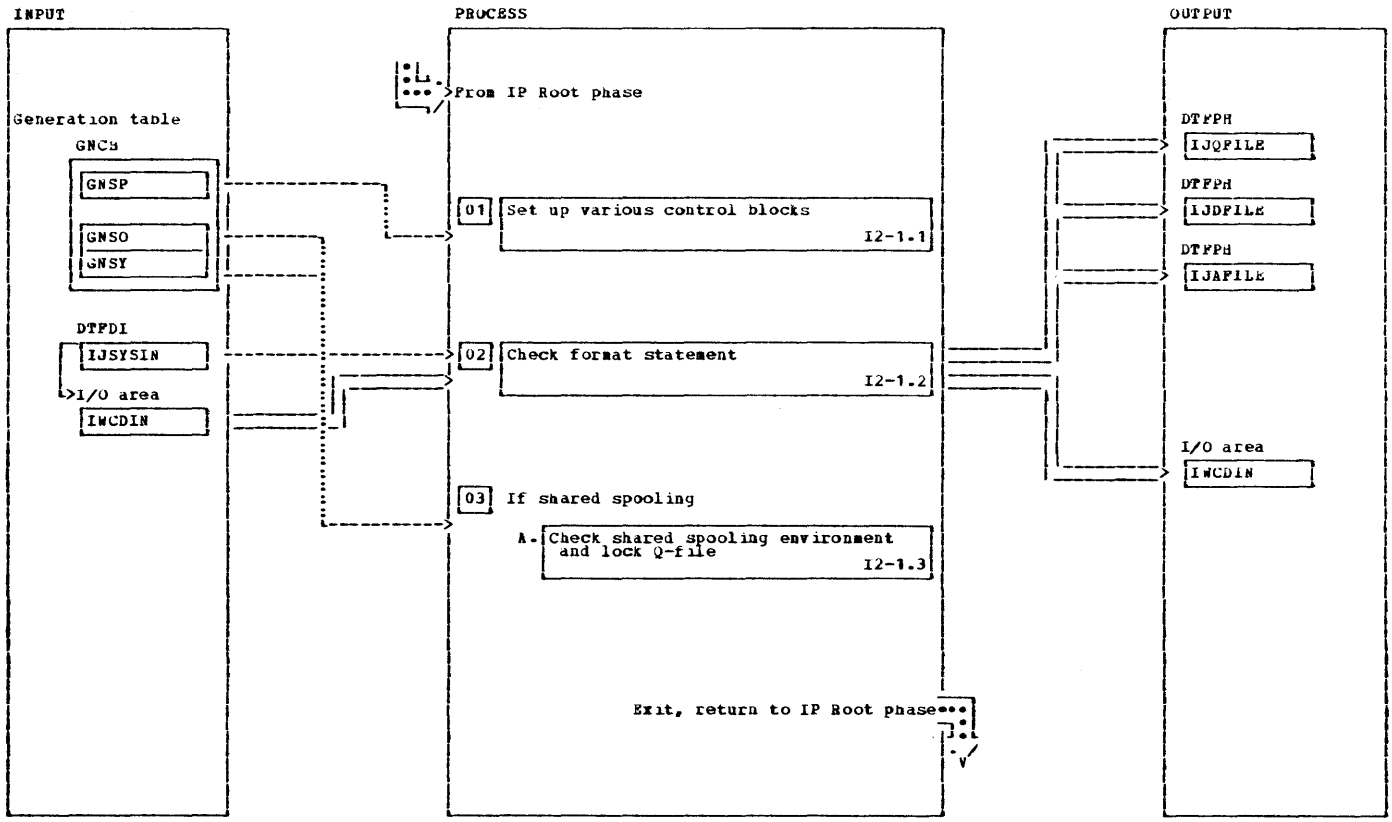


NOTES	MODULE	LABEL	REF
1	EXCP	I1CN	

NOTES	MODULE	LABEL	REF
2 Error exit: Cancel job or partition.	WAIT CANCEL		



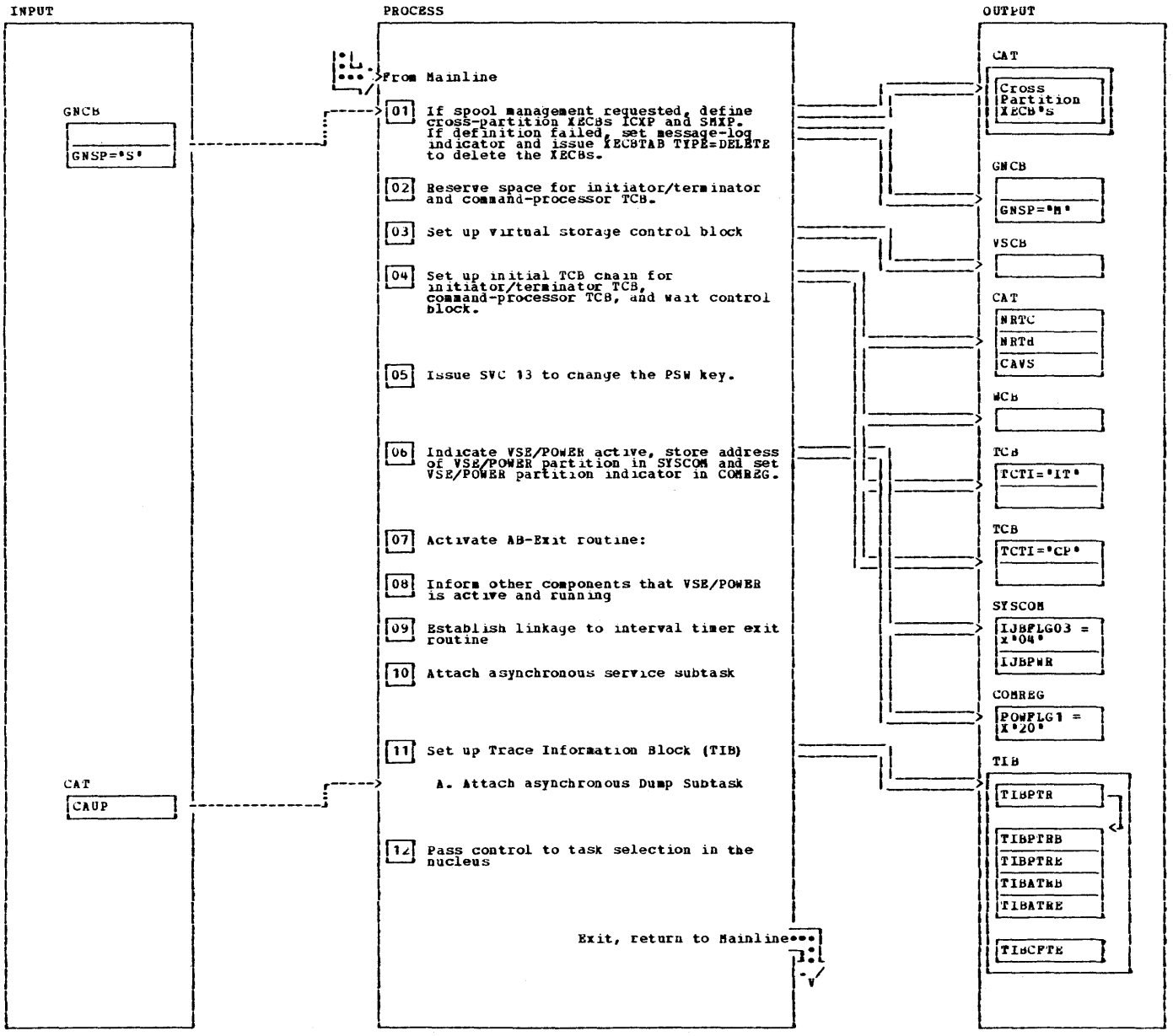
Initialize VSE/POWER



NOTES	MODULE	LABEL	REF
1 From this point on, the initialization routine is executed as a VSE/POWER internal task within the VSE/POWER partition. This is done to make use of functions and services provided by the VSE/POWER nucleus.			

NOTES	MODULE	LABEL	REF

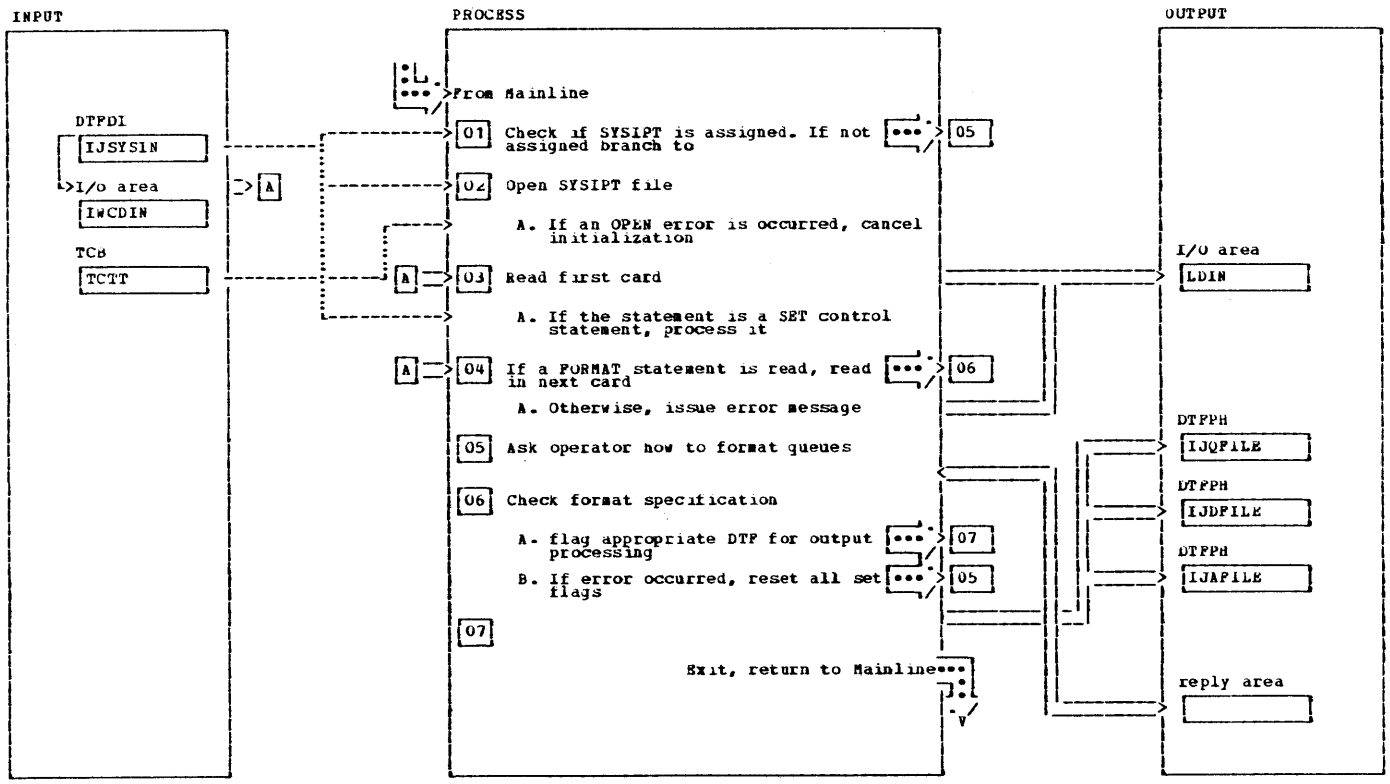
IPW$$I2 - Set up various control blocks



NOTES	MODULE	LABEL	REF
1 The macro XECBTAB TYPE=DEFINE is coded for both XECB's because the relocation of the XECB address must be done by IPW$$I2.	SVC92	I210	
	XECBTAB		$GAM
2 Storage for the initiator/terminator task and permanent command-processor task TCB are reserved		I220	$RSW
3 Storage is acquired for the virtual storage control block and initialized with following values: - storage descriptor - subpool identifiers.			$RSW
4 The initial Task Control Block (TCB, address in register 11) chain 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). The current = maximum number of tasks (2) is saved in the applicable areas.		I230	
5 A SVC 13 is issued to set the PSW key to zero. VSE/POWER will run with a key of zero in order to allow updates to supervisor control blocks and to make modifications in the serviced partitions.	SVC13		

NOTES	MODULE	LABEL	REF
8 A SUBSID macro is issued in order to inform the supervisor as well as other components that VSE/POWER is active when a bad return code has been returned, message 1QB5I INTERNAL MACRO CALL FAILED, RC=rrmm, is issued and the VSE/POWER initialization is cancelled	SUBSID		$GAM
9	STXIT		
10 The asynchronous service subtask is attached, it will be alive as long as VSE/POWER is active			$IAS
11 The Addresses and Pointers to primary and alternate Trace Area are initialized.			$RSW
11A The asynchronous Dump Subtask is attached via IPW$IAS TYPE=ATTACH,TASK=DUMP if the UPSI bit is set in CAT. It will be active as long as VSE/POWER is active.			$IAS
12 Task selection is entered which finds the initiator/terminator to be the only dispatchable task. Thus control is given back to the initiator/terminator at the next instruction and initialization continues from there on as a VSE/POWER subtask.			$WFD

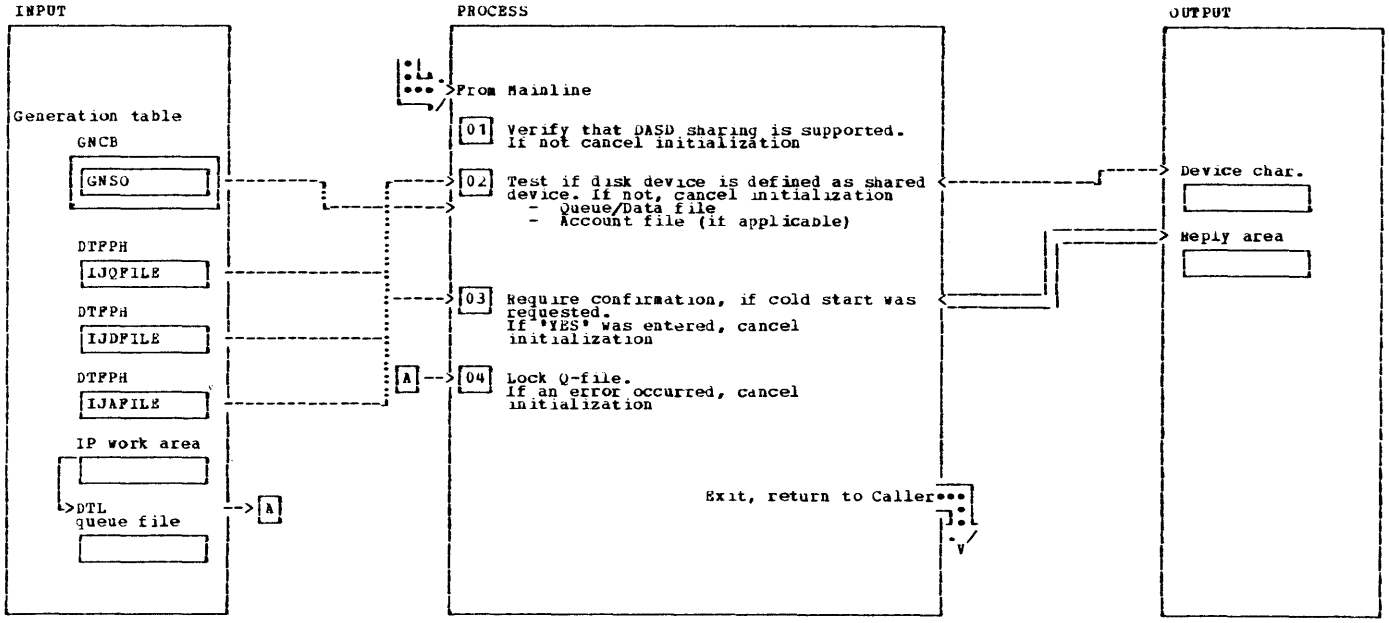
Check format statement



	NOTES	MODULE	LABEL	REF
1	The SYSIR macro is issued to obtain the appropriate PUB and LUB address of the logical unit specified in the DTFDI. The LUB is examined to see whether the logical unit (SYSIPT) is assigned or not	SYSIR	I240	
2	The SYSIPT file is opened by issuing the OPENR macro instruction	OPENR		
	If an open error is occurred, message 1Q23I LTA CANCEL is issued and the VSE/POWER initialization is terminated	CANCEL		$GAM
3	The first card is read	GET		
3A	With the SET statement a value can be assigned to one of the following keywords: - SYSID - NODE - PNET The statement is syntax checked and if invalid, the statement in error is displayed on the system console followed by message 1Q13I ERRONEOUS AUTOSTART CARD(S) READ If the SYSID statement was wrong VSE/POWER is cancelled. In all other cases, the invalid statement is ignored.			

	NOTES	MODULE	LABEL	REF
4	If the card just read was a FORMAT= statement, it is moved into a save area and the next card is read in.	GET		
4A	If the card is not a FORMAT= statement, error message 1Q13I ERRONEOUS AUTOSTART CARD(S) READ is issued			$#TO
5	If no FORMAT statement was supplied or if SYSIPT was not assigned, message 1Q11D FORMAT QUEUES= is issued to which the operator must reply NO or END/EOB, A, D, or Q.			$#TR
6	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 - queue file and data file are flagged for output NO - no data - no action (warmstart) A combination of these actions may occur. If an invalid character is found or if commas are missing, message 1Q11D is re-issued and the output flags are reset.		I2FC	

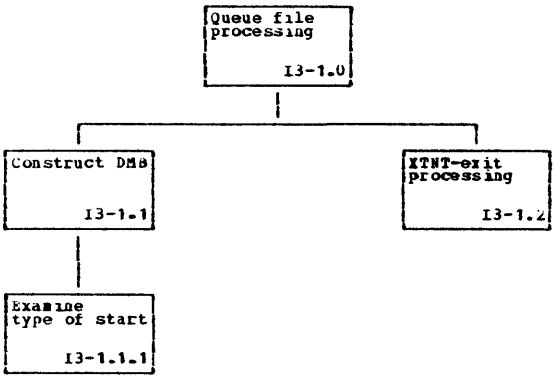
Check shared spooling environment and lock Queue-file



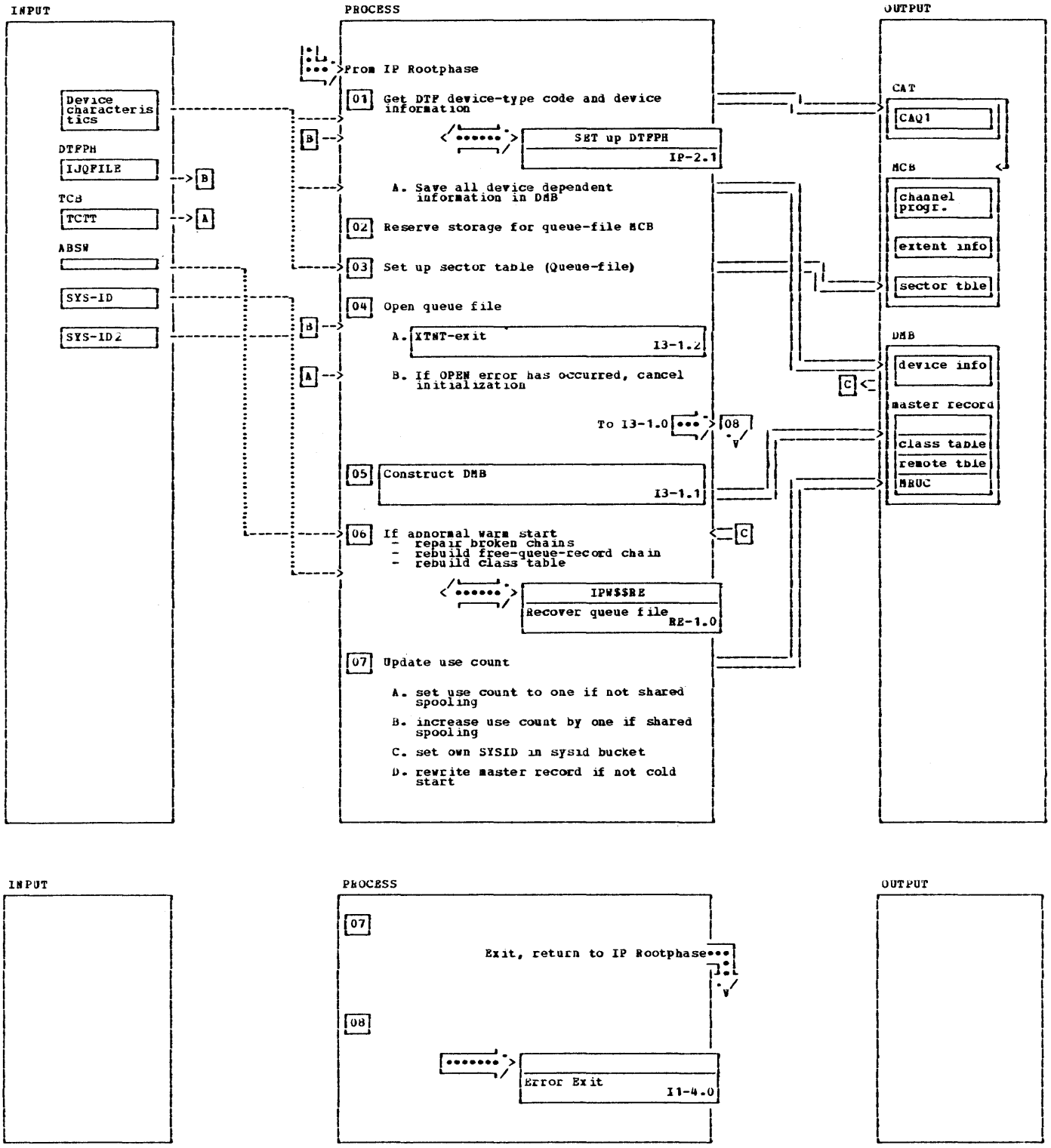
NOTES	MODULE	LABEL	REF
1 The SUBSID macro instruction is issued to check if the supervisor is generated with the DASD sharing feature. If not, message 1Q80I SUPERVISOR WITHOUT DASD SHARING FEATURE is issued and VSE/POWER initialization is terminated.	SUBSID	I250	
When a bad return code has been returned, message 1Q85I INTERNAL MACRO CALL FAILED, RC=rrmm, is issued and the VSE/POWER initialization is cancelled	CANCEL		$GAM $GAM
2 A GETVCE macro is issued in order to check if the device is defined as shared - Queue/Data file and - Account file	GETVCE		
If not, message 1Q81I filename NOT ON SHARED DEVICE is issued and VSE/POWER initialization is terminated	CANCEL		$GAM
When a bad return code has been returned, message 1Q85I INTERNAL MACRO CALL FAILED, RC=rrmm, is issued and the VSE/POWER initialization is cancelled			$GAM

NOTES	MODULE	LABEL	REF
3 If cold start for queue file and/or account file was requested, ask operator for confirmation. Message 1Q82D IS ANY OTHER VSE/POWER SYSTEM ALREADY INITIALIZED is issued to which the operator must reply 'NO' or 'YES'. If none of them is answered, message is re-issued.			$WTR
3 If the operator replied 'YES', VSE/POWER is terminated			
4 The queue file must be acquired for exclusive use by this CPU. This is done by issuing a LOCK request for the queue file. Return from the lock is only received when the lock is complete or when an error has occurred. In the later case message 1Q85I INTERNAL MACRO CALL FAILED RC=rrmm is issued and VSE/POWER initialization is cancelled.	CANCEL LOCK		$GAM
	CANCEL		$GAM

This page was left blank intentionally.



Queue file initialization

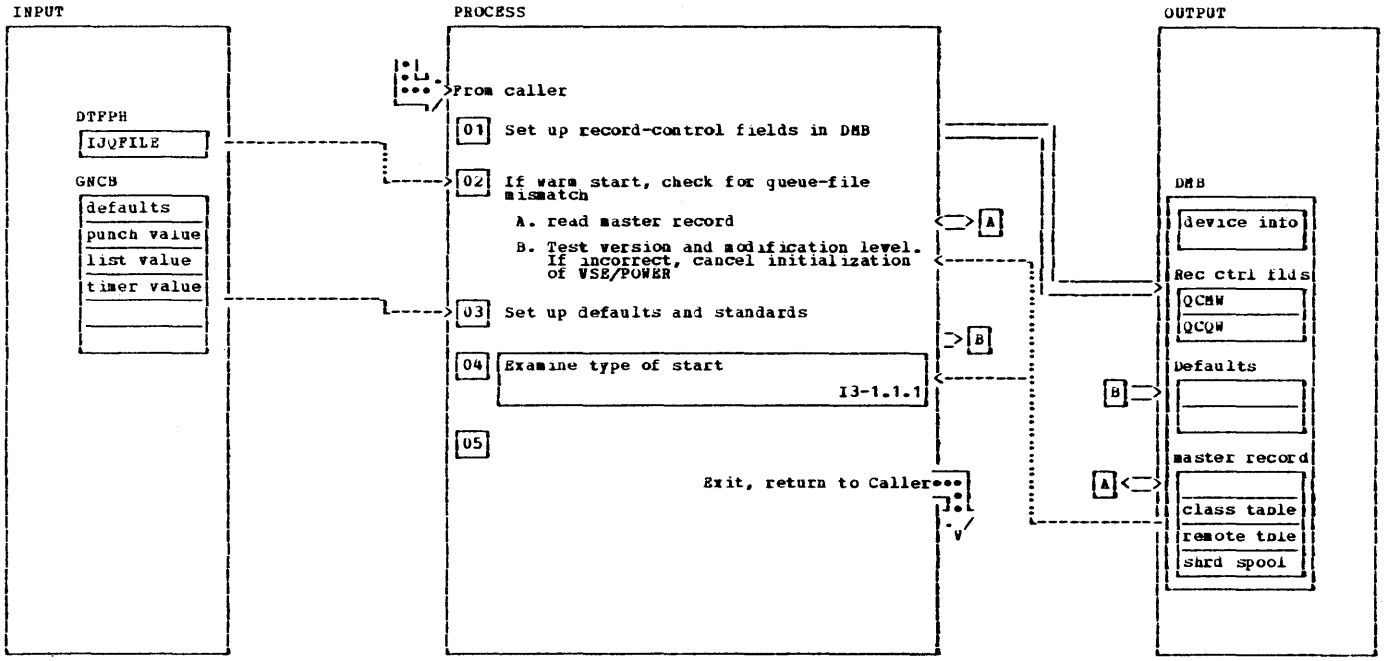


Queue file initialization

	NOTES	MODULE	LABEL	REF
	**********			
	The queue file is opened for input or output depending on the operator's specification. All relevant fields in the DMB and the appropriate MCB are initialized.			
	For warm start of the queue file, the class table is read from the master queue record.			
	For abnormal warm start, the free chain and the class chains are rebuilt properly prior to reconstructing the class table.			
	**********			
1	SET UP DTFPH routine returns in register 2 the address of the device information area of the corresponding device.			
	All device-dependent information are saved in the DMB: - # of tracks/cylinder - # of records/track	GETVCE		
2	Storage for the queue-file MCB is reserved and the address is stored in the CAT.			$RSW
	If no storage is available, message 1003I INSUFFICIENT REAL/PFIXED STORAGE ALLOCATED is issued and VSE/POWER initiation is terminated		QF00	$GAM

	NOTES	MODULE	LABEL	REF
	When the DASD device supports RPS, the sector table is built in the extension area of the MCB.	SECTVA		$CNC
4	If an error occurred during OPEN processing, the supervisor's Cancel exit posts the appropriate VSE/POWER task and sets Cancel code 'U'. Message 1023 LTA CANCEL is issued and the VSE/POWER initialization is terminated	OPENN	QF09	$GAM
6	The queue-file-recovery program is invoked to repair possible broken class chains. - reset queue set(s) which are marked in execution state - repair broken class chains - delete incomplete queue set(s) = build free-queue-record chain = restruct class table		AB00	$CNC
7	Additionally set sysid in DMB to indicate that this system owns the queue file, and rewrite the master record if not cold start of queue file		UU00	$WTQ

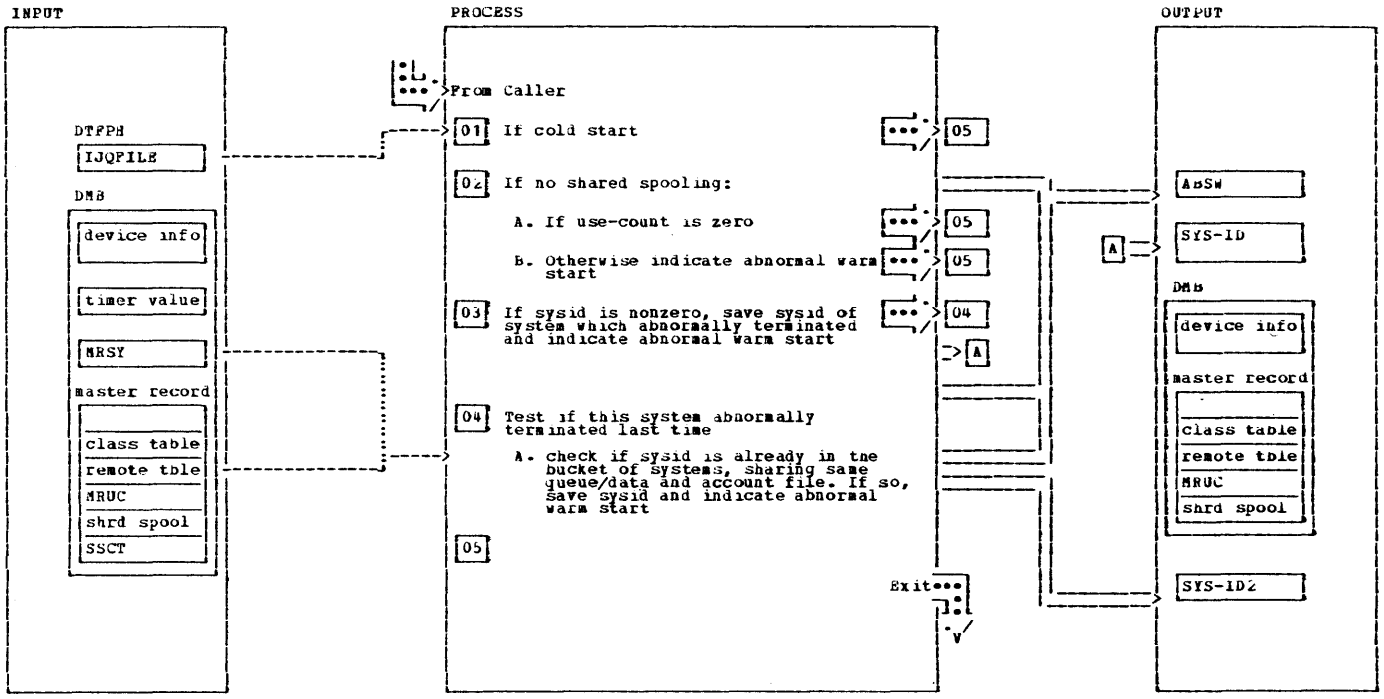
Construct disk management block (DMB)



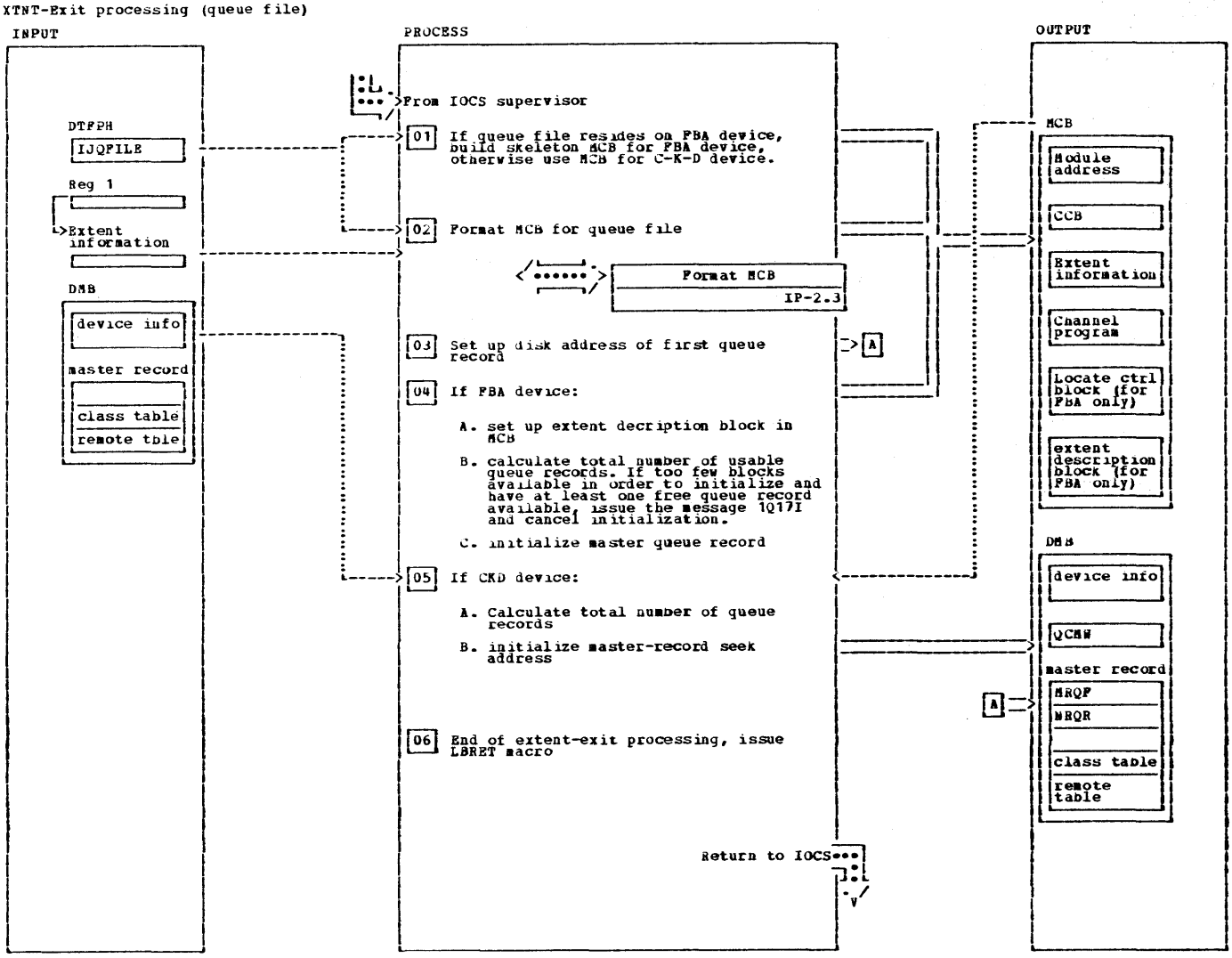
NOTES	MODULE	LABEL	REF
1 When running in /370 mode, the real address of the master-record area and the auxiliary-queue-record area are obtained and stored in the DMB	REALAD		
2B In the case of a mismatch, issue message 1Q04I QUEUE FILE MISMATCH and terminate VSE/POWER initialization			$RDQ $GAM
3 The VSE/POWER communication area, which contains default and standard values, is set up. Values are: - date and time - sublibrary - account option - default priority and options - master list and punch values - timer values - shared-spooling options - system id		QF20	$CNC

NOTES	MODULE	LABEL	REF
The shared-spooling option, if set at VSE/POWER generation time, is also set in the master queue record			

Examine type of start



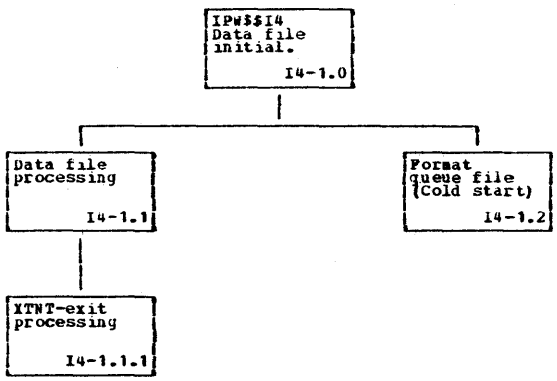
XTNT-Exit processing (queue file)

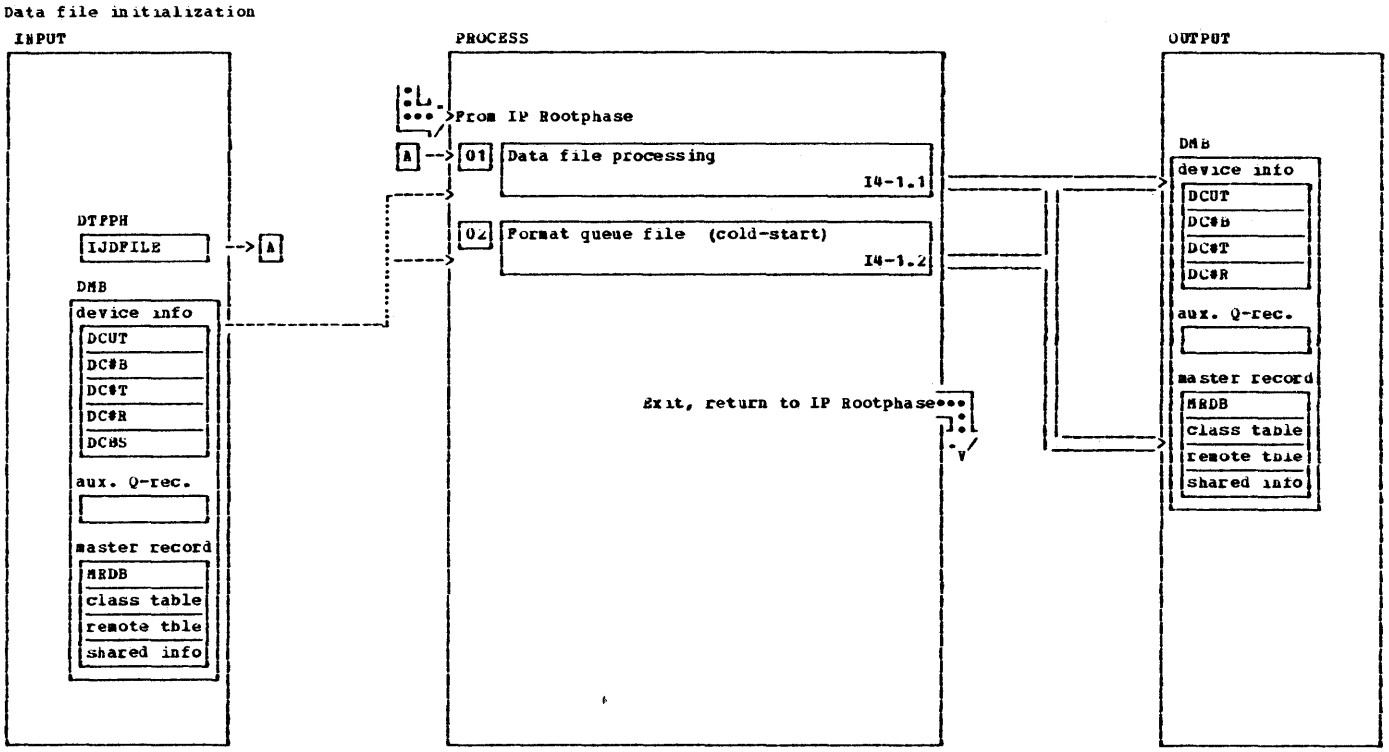


XTNT-Exit processing (queue file)

NOTES	MODULE	LABEL	REF
Control has been passed to this routine by the open processor		13ix00	
1 The skeletal queue-file MCB (either CKD of FBA version) is moved to the prior acquired storage area. The storage descriptor of the queue-file MCB is set up			
3 At this point, cold start of the queue file is assumed and the seek address of the first queue record (free-queue-record pointer) is calculated For FBA device it is '1', for CKD it is the lower-extent-limit address with record number = 2 Note: the first queue record is used for internal purposes			
4A The extent description block is initiated with following values: - physical starting block no - relative starting block no - relative ending block no - flags (permit write)			
4B The relative lower and upper limit are calulated and from these values the total number of queue entries is calculated, the first and dummy and master record blocks are subtracted, and the result stored into the 'number-of-usable-queue-records' field. If queue file too small, issue message: 1Q17I QUEUE FILE TOO SMALL			

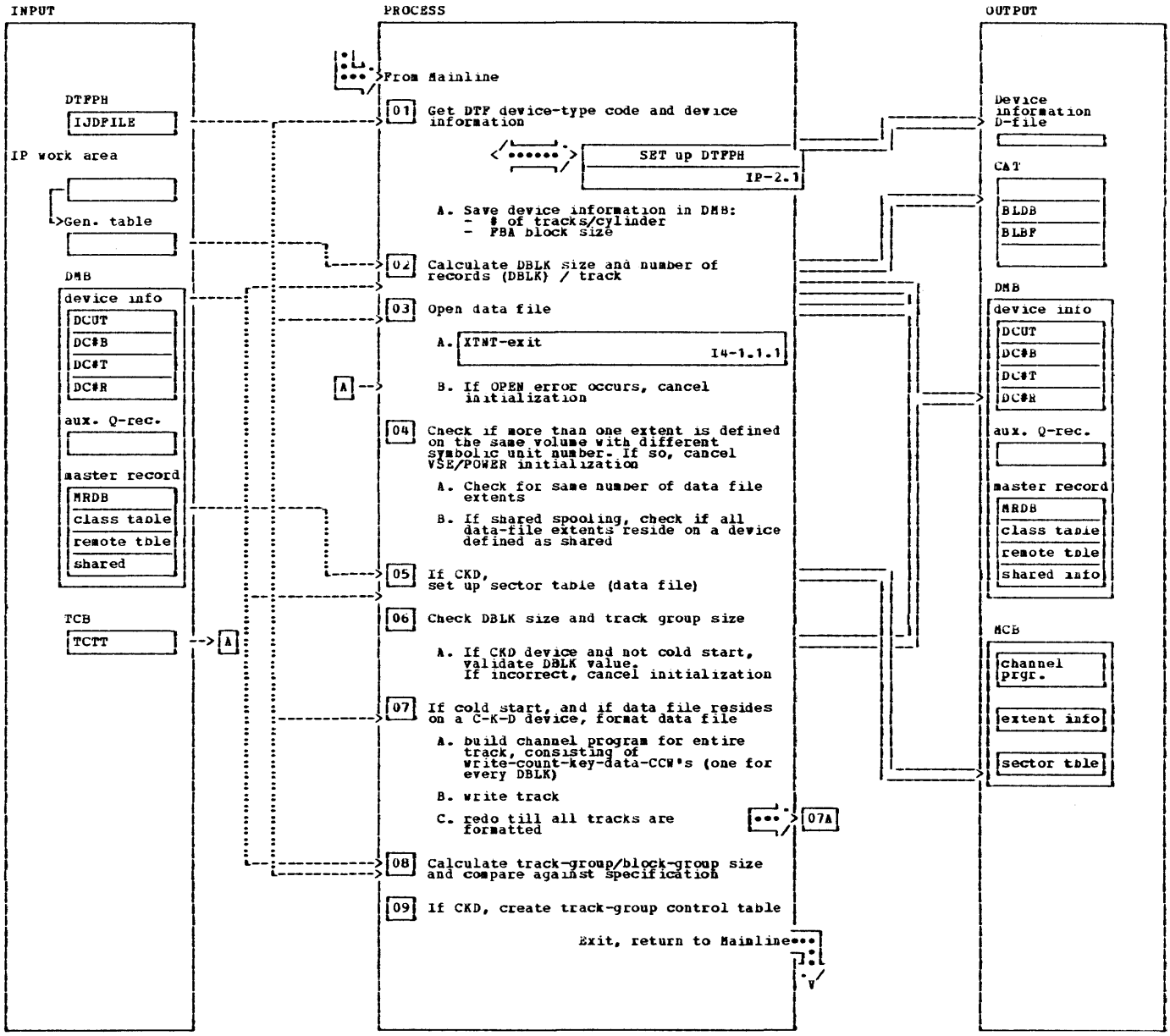
NOTES	MODULE	LABEL	REF
5A The relative lower and upper limit are calculated and from these values the total number of queue entries is calculated and stored into the 'number-of-queue-records' field			
5 Calculation of queue records: for FBA-device - Higher limit of extent (MFHI), - minus lower limit of extent (MFLO), - minus master record, - minus dummy record. for C-K-D device - calculate number of tracks for lower limit - calculate number of tracks for higher limit - calculate number of records. The master record, which is longer than a normal queue record, is placed on the end of the queue-file extent			
6 Control is returned to the calling routine via a LBRET 1 macro instruction	LBRET		




Data file initialization
INPUT
PROCESS
OUTPUT
From IP Rootphase
A
01
Data file processing
I4-1.1
02
Format queue file (cold-start)
I4-1.2
DTFPH
IJDFILE
A
DMB
device info
DCUT
DC$B
DC$T
DC$R
DC$S
aux. Q-rec.
master record
MRDB
class table
remote tble
shared info
Exit, return to IP Rootphase
DMB
device info
DCUT
DC$B
DC$T
DC$R
aux. Q-rec.
master record
MRDB
class table
remote tble
shared info

This page was left blank intentionally.

Data file initialization

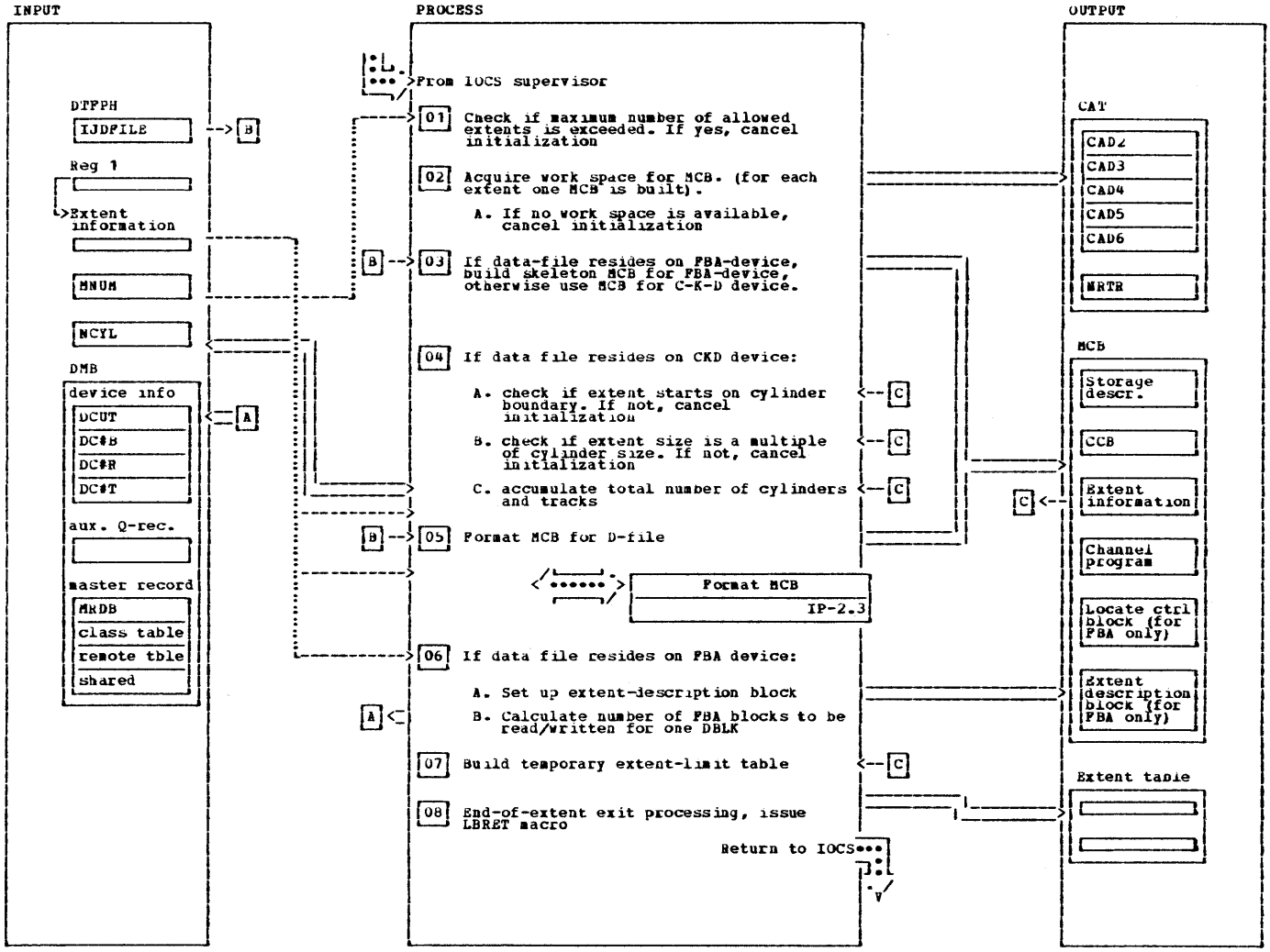


Data file initialization

	NOTES	MODULE	LABEL	REF
	**********			
	The data file is opened for input or output depending on the format of the operator's command. All relevant fields in the DMB and the appropriate MCb(s) are initialized.			
	If the data file resides on a C-K-D device and cold start was requested, the data file is formatted with fixed-length records (length=DBLK) containing hex zeros.			
	If the data file resides on an FBA device(s), no formatting will be done.			
	**********			
1	SET up DTFPH routine returns in register 2 the address of the device-information area of the appropriate device and register 1 contains the default DBLK size		DF00	
2	The DBLK value obtained from the master record or the generation table is rounded up to a multiple of 32. The number of blocks/track is calculated and saved in the DMB	GETVCE		
3	The data file is opened by issuing the OPENR macro instruction	OPENR	DF09	
3B	If an error occurs during OPEN processing, the supervisor cancel exit posts the appropriate VSE/POWER task and sets cancel code 'U'. Message 1Q23 LTA CANCEL is issued and the VSE/POWER initialization is terminated			$GAM
				$CNC
4	The created MCBs are checked if more than one extent is defined on the same volume. If so the symbolic unit numbers must be the same. If the extents are on different volumes, the logical units must be in ascending sequence. Issue message 1Q191 INVALID EXTENTS IJDFILE and cancel VSE/POWER initialization if this is not the case.		DF12	$GAM
4A	When warm start of the data file is performed, the number of data file extents must be the same as at cold-start time. If not, message 1Q191 'INVALID EXTENTS IJDFILE is issued and VSE/POWER initialization is cancelled	CANCEL		$GAM
4B	If shared spooling, all data file extents which do not reside on the same volume as the first data file extent are checked to see if they reside on the shared disk device. If not, message 1Q81I 'filename' NOT ON SHARED DEVICE is issued and the VSE/POWER intialization is cancelled		DF15	$GAM

	NOTES	MODULE	LABEL	REF
	This is done by issuing the GETVCE macro.	GETVCE		
	If a bad returncode (¬=0) is given back, message 1Q85I INTERNAL MACRO CALL FAILED, RC=rrmm is issued and VSE/POWER initialzation is terminated			$GAM
5	If RPS is supported, the sector table is built in the MCB	SECTVAL	DF18	
6A	If warm start or abnormal warm start, the DBLK value is checked for accurancy. This is done by reading count field of the first data block and comparing it against the DBLK value. If it does not match, message 1Q04I QUEUE FILE MISMATCH is issued and VSE/POWER initialization is cancelled	EXCP		$WPC
		CANCEL		$GAM
7	A channel program is built which writes a track of records, each record (DBLK) bytes long. This channel program is issued for each track on all extents of the data file	EXCP	FDCKD	$WPC
8	The number of track groups/block groups is calculated in such a way that maximum spooling advantage is obtained (1-to-1 ratio queue and data file if possible). This value is checked against the number of tracks/cylinder of the device and the TRACKGP specification or against the BLOCKGP specification, if FBA. Depending on the values calculated, one or two of following messages might be printed:		DF48	
	1Q17I QUEUE FILE TO SMALL 1Q09I TRACK GROUP CHANGED TO nn 1Q0BI BLOCK GROUP SIZE CHANGED OR SET TO nn 1Q0CI QUEUE FILE TOO LARGE, USED=nnnnn 1Q0FI DATA FILE SPECIFICATION ERROR this value is compared against the specification done at the VSE/POWER generation time.			$GAM
9	A track-group control table is built, one byte per track. The byte contains (head address + 1) of the corresponding track, or for last-in-track group or non-available tracks, x'00'		DF76	

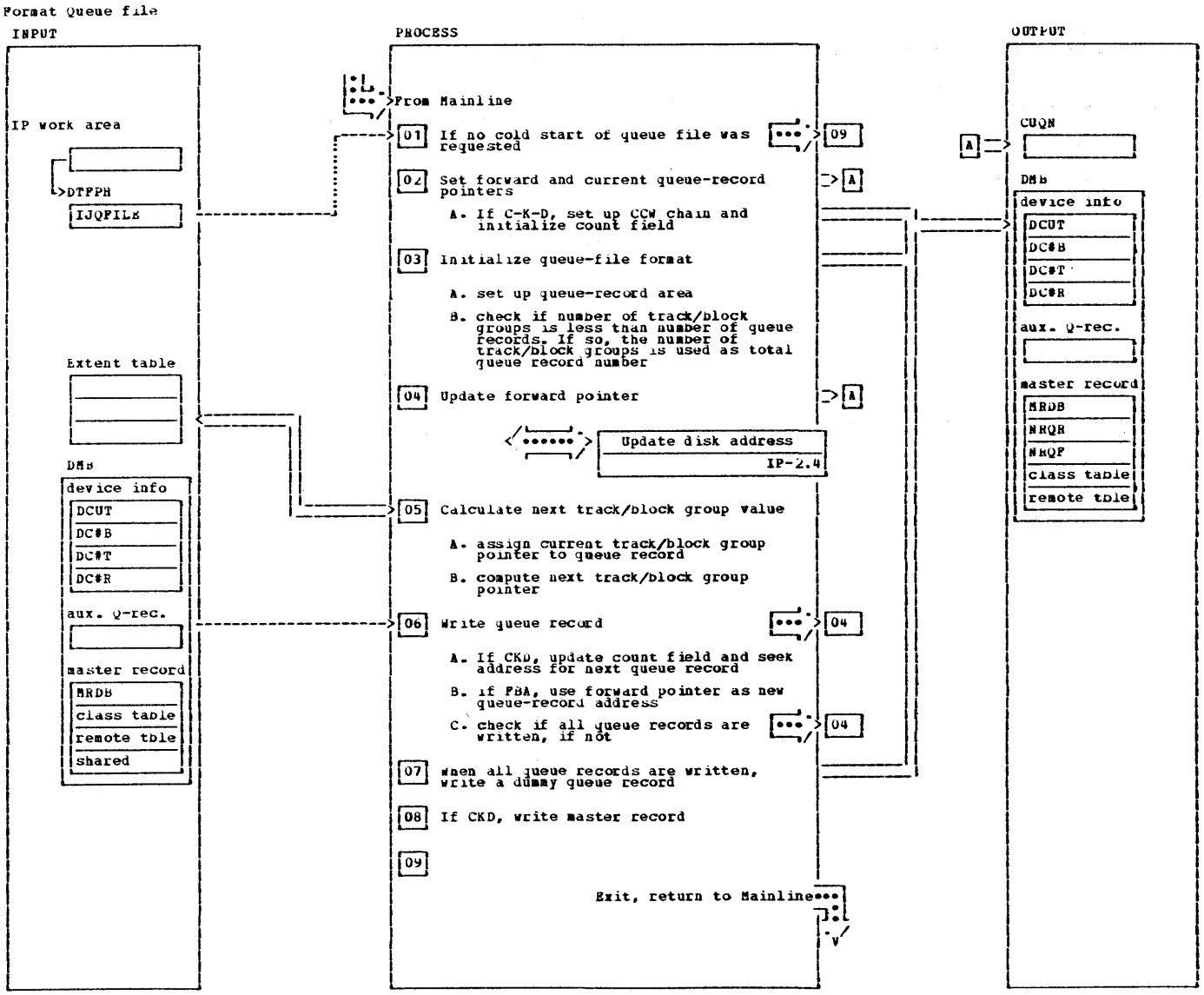
ITNT-Exit processing (Data file)



NOTES	MODULE	LABEL	REF
Control has been passed to this routine by the open processor		I4DX	
1 If more than 5 extents are specified, message 1Q18I TOO MANY EXTENTS IJDFILE is issued and VSE/POWER initialization is terminated	CANCEL		$GAM
2 Storage for the MCB is acquired			$RSW
If no work space is available, message 1Q03I INSUFFICIENT REAL/FIXED STORAGE ALLOCATED is issued and VSE/POWER initialization is terminated. Otherwise the address of the MCB is stored in the CAT	CANCEL		$GAM
3 The skeletal data file MCB (either CKD or FBA version) is moved to the just-acquired storage area. The module index number is updated and converted into printable format and stored in the storage descriptor of the MCB			
4 If the lower track number (head) is not zero, the extent does not start on cylinder boundary. If so, message 1Q19I INVALID EXTENT IJDFILE is issued and VSE/POWER initialization is terminated			

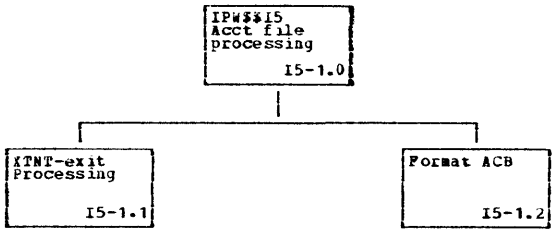
NOTES	MODULE	LABEL	REF
4C The total number of cylinders and the total number of tracks are calculated	CANCEL		$GAM
6 The DBLK size is stored tne tne read/write CCW.			
6A Values initiated are: - physical starting block no - relative starting block no - relative ending block no - flags (permit write)			
6B The number of FBA blocks occupied by one data block is calculated and is stored the locate CCW of the FBA channel program			
7 For each extent one entry in the extent limit table is created. An entry contains following information: - index to MCB - rel. starting block number or CCHHR for CKD - rel. ending block number or CCHHR for CKD			
8 Control is passed back to OPEN via the LBRET Z macro instruction	LBRET	DI18	

This page was left blank intentionally.

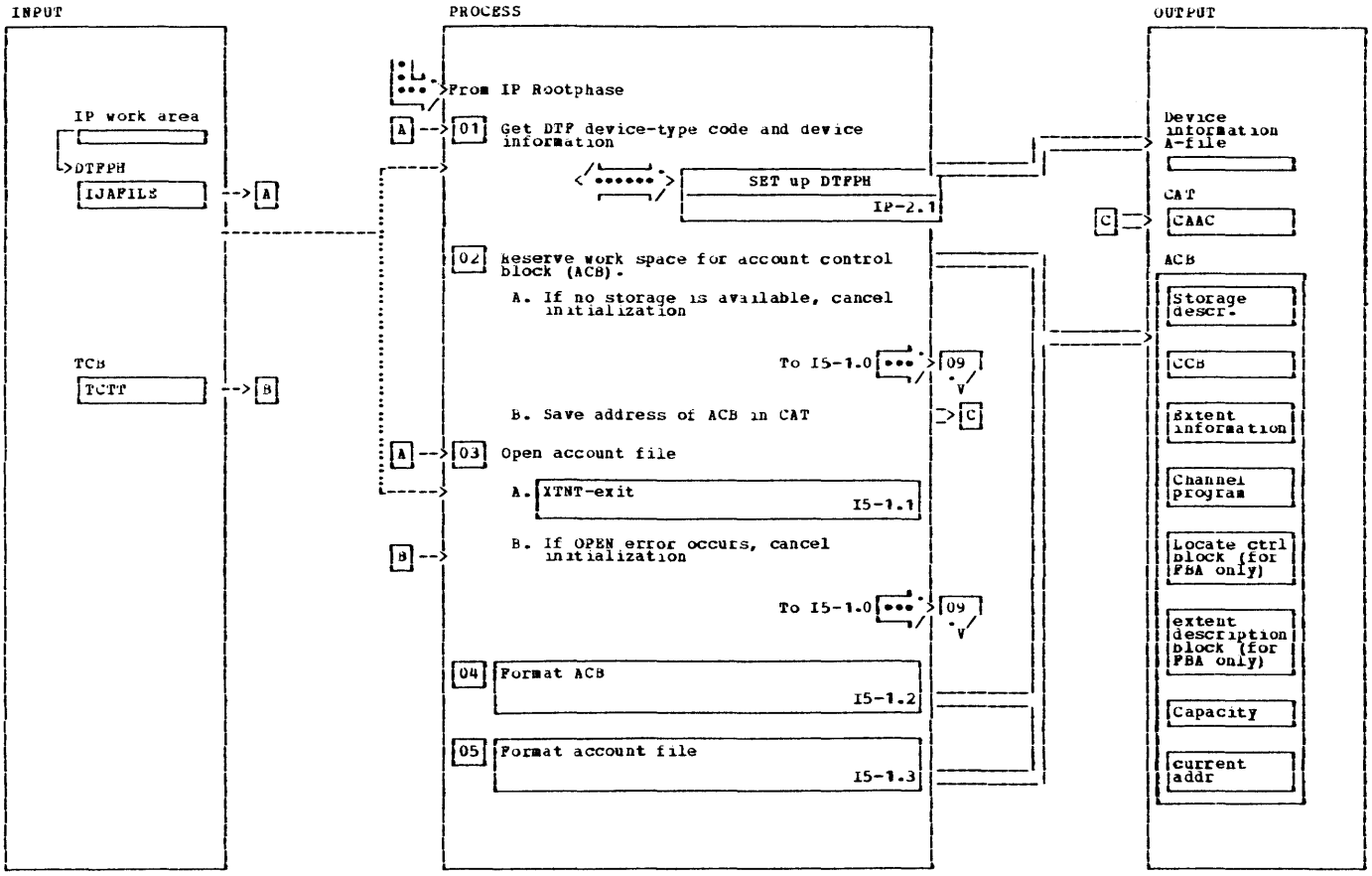

Format Queue file
INPUT
PROCESS
OUTPUT
IP work area
DTFPH
IJQFILE
Extent table
DMB
device info
DCUT
DC#B
DC#T
DC#R
aux. Q-rec.
master record
MRDB
class table
remote tble
shared
From Mainline
01 If no cold start of queue file was requested
09
02 Set forward and current queue-record pointers
A
A. If C-K-D, set up CCW chain and initialize count field
03 Initialize queue-file format
A. set up queue-record area
B. check if number of track/block groups is less than number of queue records. If so, the number of track/block groups is used as total queue record number
04 Update forward pointer
A
Update disk address
IP-2.4
05 Calculate next track/block group value
A. assign current track/block group pointer to queue record
B. compute next track/block group pointer
06 Write queue record
04
A. If CKD, update count field and seek address for next queue record
B. if FBA, use forward pointer as new queue-record address
C. check if all queue records are written, if not
04
07 When all queue records are written, write a dummy queue record
08 If CKD, write master record
09
Exit, return to Mainline
A
CUQN
DMB
device info
DCUT
DC#B
DC#T
DC#R
aux. Q-rec.
master record
MRDB
NRQR
NRQF
class table
remote tble

Format Queue file

	NOTES	MODULE	LABEL	REF
	**********			
	The queue file is formated only when cold start is performed.			
	**********			
1	If no cold start of queue file was requested, branch to exit		I4FQ	
2	If FBA device, address 2nd queue record and initialize forward pointer The first queue record is used for internal purposes only			
2A	A CCW chain is set up, consisting of: - Seek - TIC or Set sector if applicable - Write count - Write data to address the first queue record to be written.			
3	Set queue record to hex zeros and indicate free queue record. Process all track/block groups and set last to x'00'. Check if total number of queue records is acceptable. If not, set it to the value of track/block groups.			
4	Update the queue-file disk address and deduct one from the number of queue records to be written. When the value is zero, the forward pointer is set to zero.			
5	All track/block groups are subsequently inspected. The order in which this is done is by taking track/block groups in turn from each of the data file extents, if more than one extent is specified. The disk address, pointing to the next track/block group in the extent, is calculated and saved in the temporary extent-limit table. When an extent is exhausted, its temporary extent-limit-table entry is marked with x'FE' in its first byte.			
6	The forward pointer is inserted in the 'next record in set' field. Then the queue record is written	EXCP		$wTQ
7		EXCP		$wPC
8		EXCP		$wPC



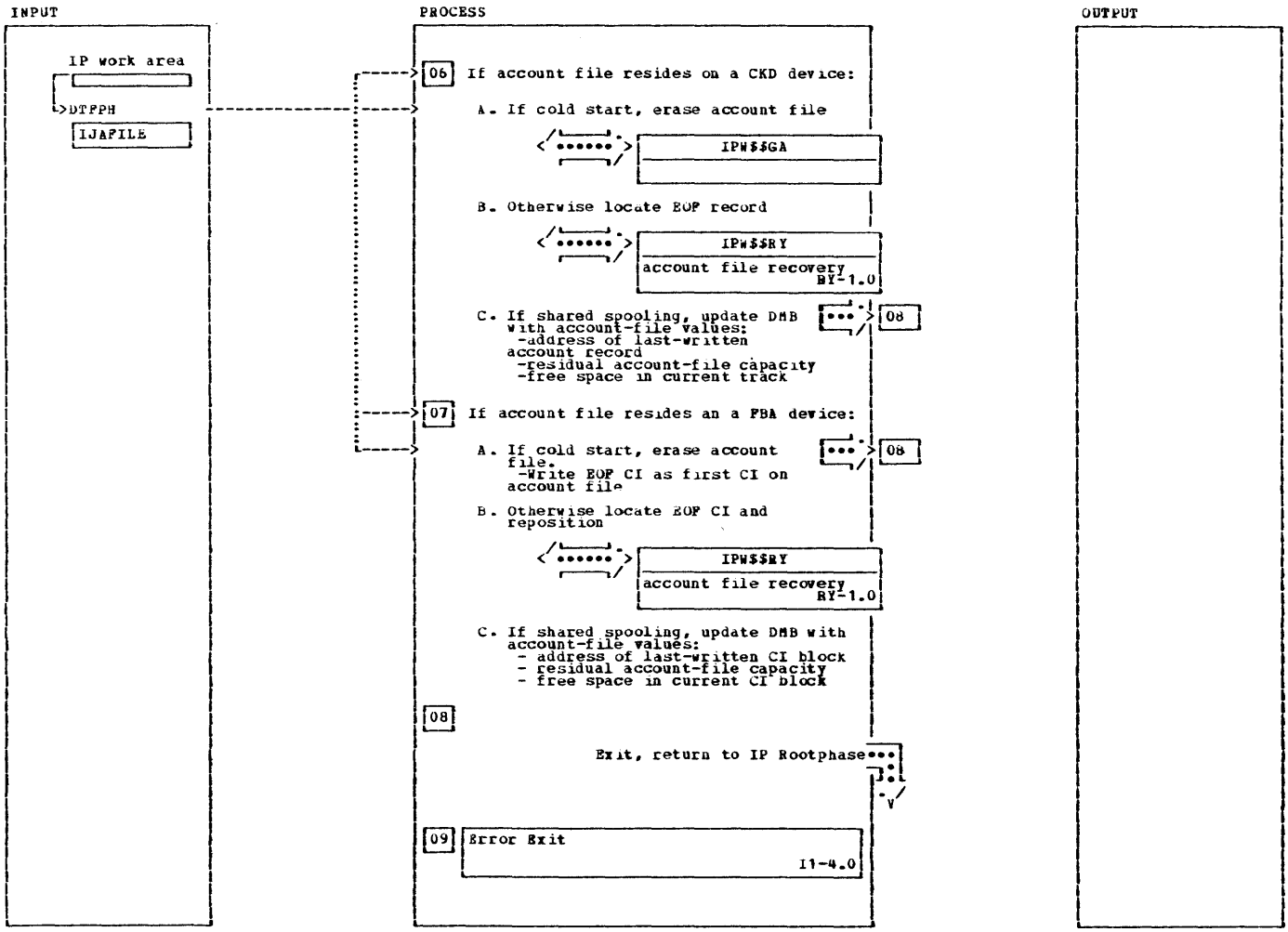
Account file initialization



NOTES	MODULE	LABEL	REF
**********			
The account file is opened for input or output depending on the format specification done by the operator. All relevant fields in the DMB and the ACB are initialized. If a cold start was requested, an EOF record is written on each track of the account file when a C-K-D device is used. When an FBA device is used, an EOF CI block is written on the first FBA block. If warm start, the last-written account record is located and its disk address is saved in the ACB and DMB respectively. The residual file capacity is calculated.			
**********			

	NOTES	MODULE	LABEL	REF
1	The SET up DTFPH routine returns in register 2 the address of the device information area. The device information is saved in core for later usage.			
2	Storage for the account control block (ACB) is reserved. The address of the ACB is saved in the CAT. If not enough work space is available, message 1003I INSUFFICIENT REAL/PFIXED STORAGE ALLOCATED is issued and VSE/POWER initialization is terminated	CANCEL	AF00	$RSW $GAM
3		OPENR		
3	If an error occurs during OPEN processing, the supervisor cancel exit posts the appropriate VSE/POWER task and sets cancel code 'U'. Message 1023 LTA CANCEL is issued and the VSE/POWER initialization is terminated	CANCEL		$GAM

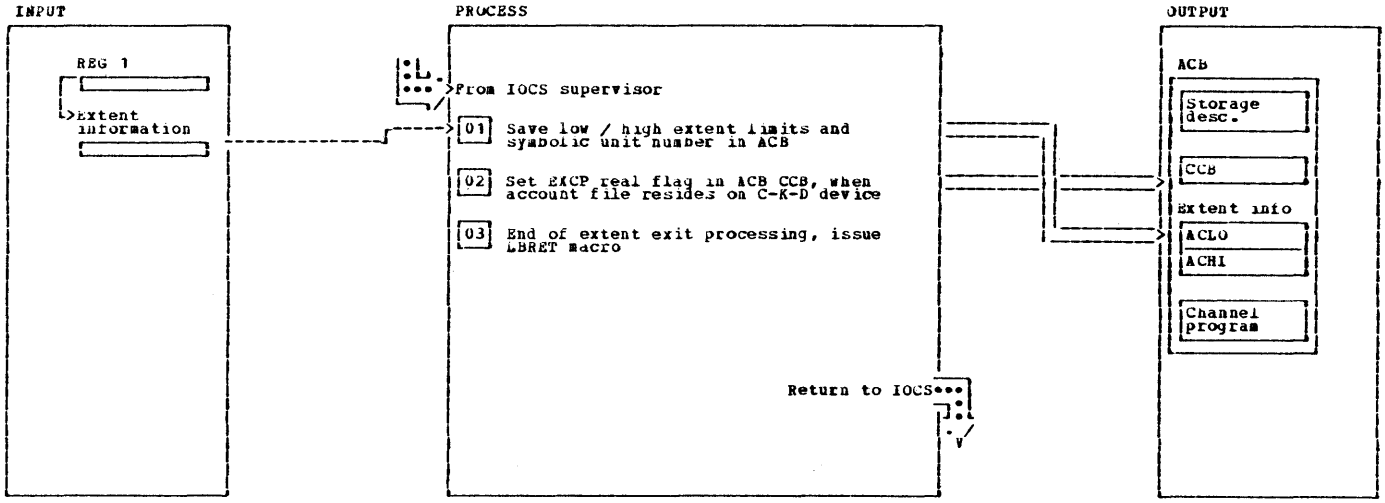
Account file initialization



	NOTES	MODULE	LABEL	REF
6A	If this account file was cold-started (PHOP byte in DTFPd set to x '80'), the account file is erased and an EOF record is written as the first record on each track.			

	NOTES	MODULE	LABEL	REF
7A	When cold start is requested, an EOF record is written as first record on the file.	EXCP		$WFC

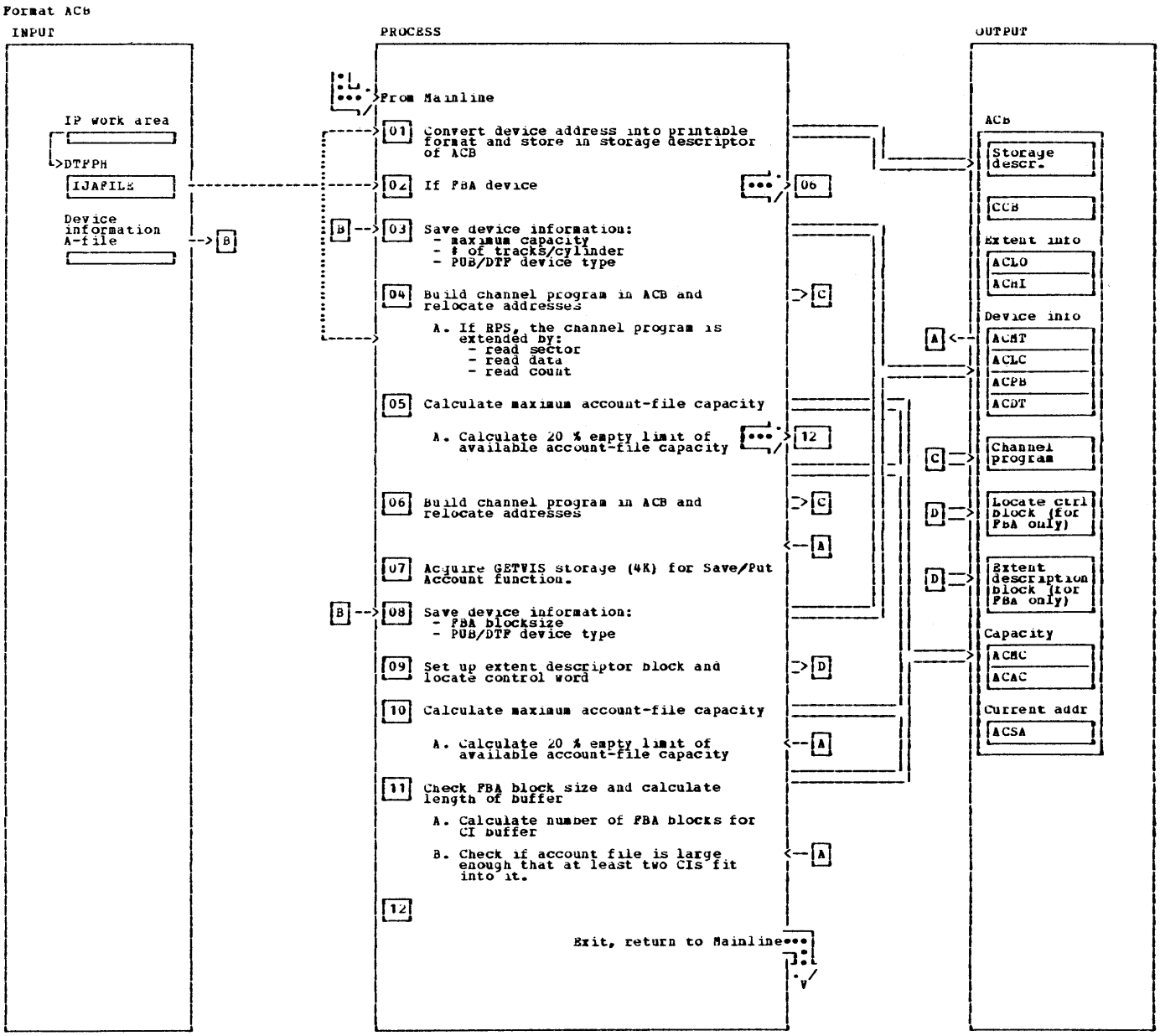
XTNT-Exit processing (account-file)



NOTES	MODULE	LABEL	REF
Control has been passed to this routine by the open processor		ISAX	

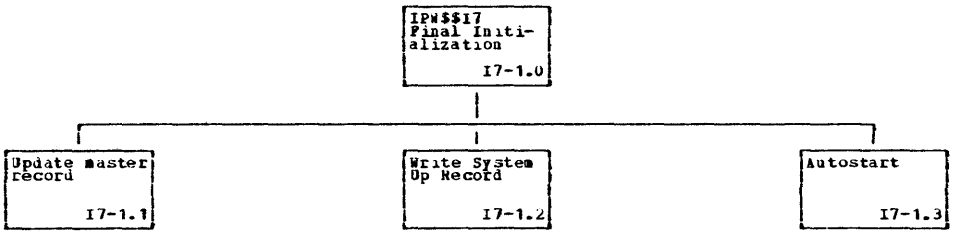
NOTES	MODULE	LABEL	REF
3 Control is returned to the calling routine via a LBRET 1 macro instruction	LBRET		

This page was left blank intentionally.

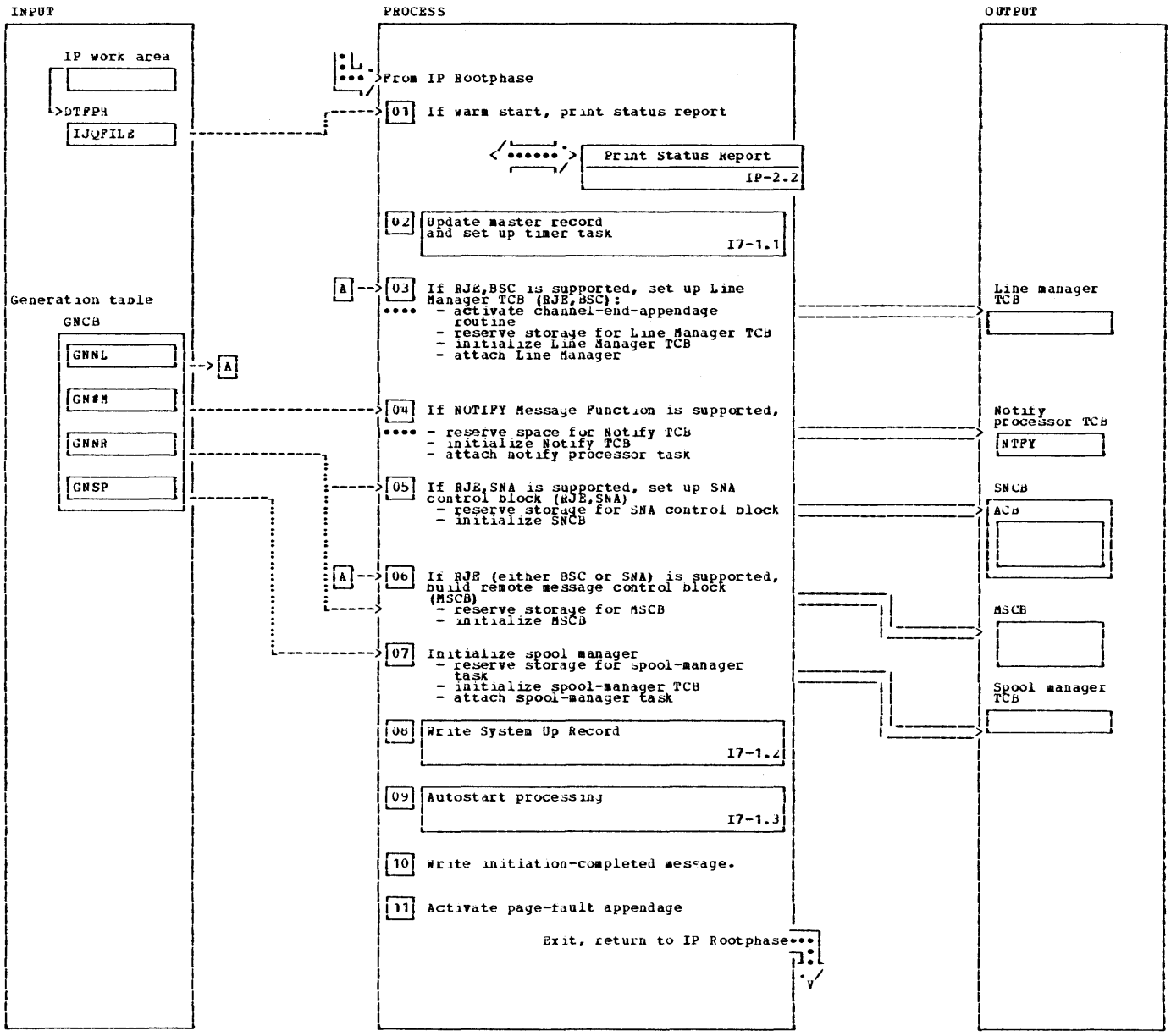

Format ACB
INPUT
PROCESS
OUTPUT
From Mainline
IP work area
DTFPH
IJAFILE
Device information A-file
01 Convert device address into printable format and store in storage descriptor of ACB
02 If FBA device
03 Save device information:
- maximum capacity
- # of tracks/cylinder
- PUB/DTF device type
04 Build channel program in ACB and relocate addresses
A. If RPS, the channel program is extended by:
- read sector
- read data
- read count
05 Calculate maximum account-file capacity
A. Calculate 20 % empty limit of available account-file capacity
06 Build channel program in ACB and relocate addresses
07 Acquire GETVIS storage (4K) for Save/Put Account function.
08 Save device information:
- FBA blocksize
- PUB/DTF device type
09 Set up extent descriptor block and locate control word
10 Calculate maximum account-file capacity
A. Calculate 20 % empty limit of available account-file capacity
11 Check FBA block size and calculate length of buffer
A. Calculate number of FBA blocks for CI buffer
B. Check if account file is large enough that at least two CIs fit into it.
12
Exit, return to Mainline
ACB
Storage descr.
CCB
Extent info
ACLO
ACHI
Device info
ACMT
ACLC
ACPB
ACDT
Channel program
Locate ctrl block (for FBA only)
Extent description block (for FBA only)
Capacity
ACMC
ACAC
Current addr
ACSA

Format ACB

	NOTES	MODULE	LABEL	REF
1	The ACB storage descriptor is initialized and set in the ACB.		ACOO	
1	An EXTRACT macro is issued to get the PUB entry that is associated with the logical unit SYS000. The device address is converted into printable format and stored in the ACB If a bad return code (¬=0) is given back, message 1Q55I INTERNAL MACRO CALL FAILED, RC=rrmm is issued and VSE/POWER is cancelled.	EXTRACT		$GAM
3	All device-dependent information is saved in the ACB: - maximum track capacity - residual track capacity - number of tracks per cylinder - PUB device-type code - DTFPH device-type code			
4	The write/read channel program in the ACB is set up. For this purpose the CCW addresses are relocated. 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. The channel program consists of - SEEK - TIC + 8 or SET SECTOR - SEARCH - TIC -8 - WRITE COUNT, KEY, DATA			
5	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 is then calculated from this value and stored in the ACB.			
6	The write/read channel program in the ACB is set up. For this purpose the CCW addresses are relocated. The channel program consists of - DEFINE EXTENT - LOCATE - READ/WRITE			
7	The first 2K of the area is used by the put account function to hold a copy of the last used CI in core. The 2nd part of the area is used by save account as input area. If the GETVIS request failed, message 1Q26I GETVIS AREA TOO SMALL is issued.	GETVIS		$GAM
8	FBA device information is saved in the ACB: - block size - PUB device-type code - DTFPH device-type code			
9	The extent-description block is initiated with following values: -physical starting block number -relative starting block number (=0) -relative ending block number -flags (permit writes)			
10	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 is then calculated from this value and stored in the ACB			
11	The blocksize for the account file is set to 2048 bytes. This value is then stored in the read/write CCW of the channel program.			
11A	The number of FBA blocks used to contain one CI buffer is calculated. This is done by dividing the CI block size (2048 bytes) by the FBA block size and rounding up to the next integer. If the account file is not large enough to contain at least 2 CIs, message 1Q0DI IJAFILE TOO SMALL, REQUIRED BLOCKS=xxx is issued and VSE/POWER initialization is terminated.	CANCEL		$GAM



IPWSSI7 - Final Initialization

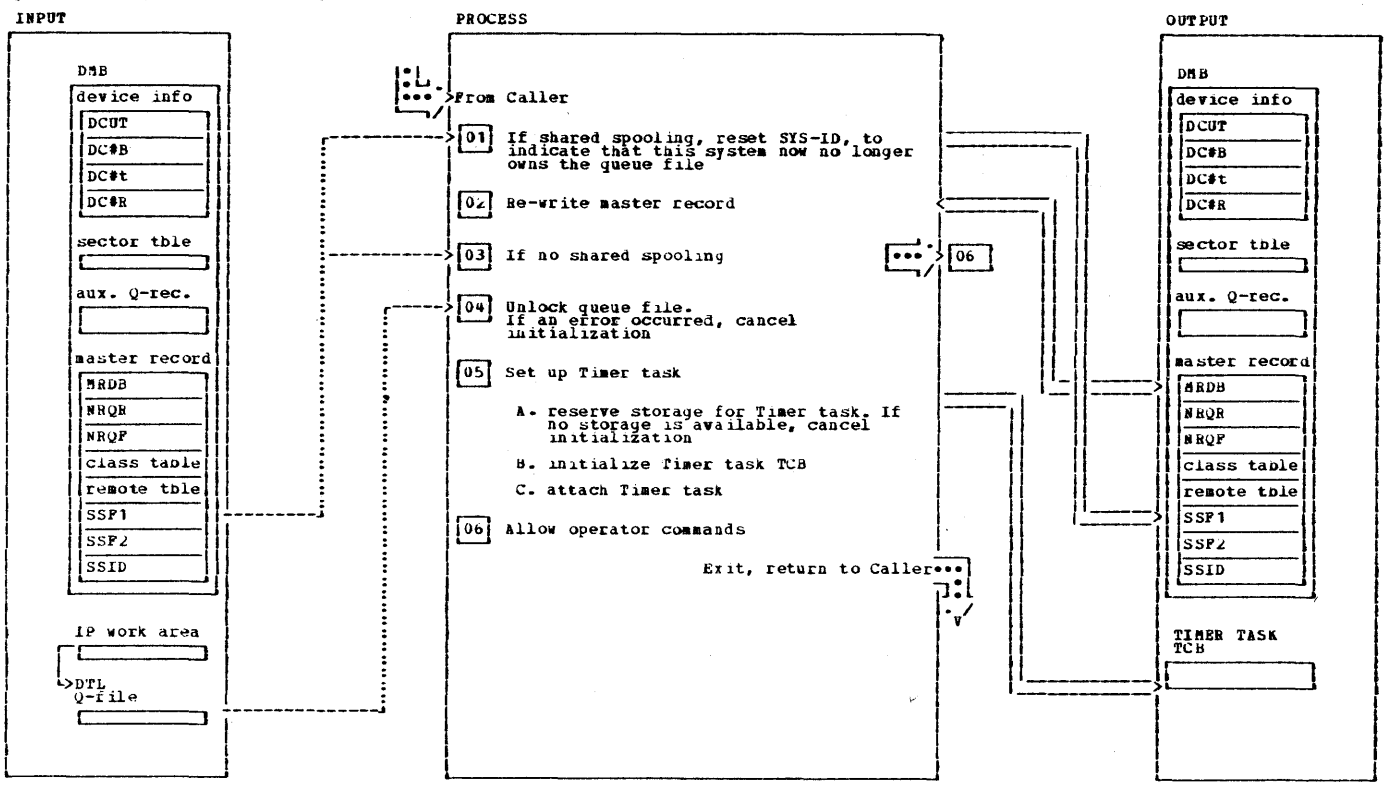


IPW$$I7 - Final Initialization

	NOTES	MODULE	LABEL	REF
1	The Status report is only printed when a warm start is performed and when SYSLST is assigned to a valid printer device			
3	The channel-end-appendage routine is activated	SETAPP	FN20	
	Storage for the line-manager task is reserved. If no storage is available message 1003I INSUFFICIENT REAL/PFIXED STORAGE ALLOCATED is issued and the VSE/POWER initialization is terminated.		FN30	$RSW
	The task control block (TCB) is initialized and the line- manager task is attached			$GAM $ATT
	After attaching the line-manager task, it is set inactive and the timer intervall routine is activated.	STXIT		
4	GN#M in POWER/VSE generation table contains the max number of messages for Notify message queue. This field is initiated at POWER/VSE initialization time when the NTFYMSG parameter has been generated.			$RSW
	Storage is reserved for the Notify processor TCB and the TCB is initialized. The Notify processor task is then attached by means of the IPW$ATT-macro instruction.			
	If we did not get workspace for the TCB we do an error report and the initialization is cancelled.		FN35	$ATT
5	Storage for the SNA control block (SNCB) is reserved		FN40	$RSW
	If no storage is available message 1003I INSUFFICIENT REAL/PFIXED STORAGE ALLOCATED is issued and the VSE/POWER initialization is terminated.			
	The address of the SNCB is saved in the CAT and the SNCB is initialized: - storage descriptor - address of SUCB space - address of compaction table pool - max. numbers of SU's - address RMCB - address of logon space - ACB image			$GAM

	NOTES	MODULE	LABEL	REF
	The logon/logoff GETVIS area, which was acquired previously by IPW$$I1, is released	FREEVIS		
	Storage for the remote message control block is reserved. If no storage is available, message 1003I INSUFFICIENT REAL/PFIXED STORAGE ALLOCATED is issued and the VSE/POWER initialization is terminated.		FN50	$RSW $GAM
7	If spool management was requested via SPOOL=YES option in the POWER generation macro, the in-core reader and the spool/command task are initialized		FN60	
	Storage for the spool-manager task is reserved. If no storage is available message 1003I INSUFFICIENT REAL/PFIXED STORAGE ALLOCATED is issued and the VSE/POWER initialization is terminated.			$RSW
	The TCB is initiated with following values: - descriptor - task-save-area address - TCB address - address of cross-partition XECB's			
	Finally the task is attached			$ATT
8			FN70	
9	The autostart routine is invoked, unless termination is requested by one of theVSE/POWER features.		FN70	
10	Message: 1021I VSE/POWER INITIATION IS COMPLETED		FN80	$WTO
11	The page-fault-appendage processing is activated by issuing the SETPFA macro instruction. From now on VSE/POWER handles all page fault which occur.	SETPFA		

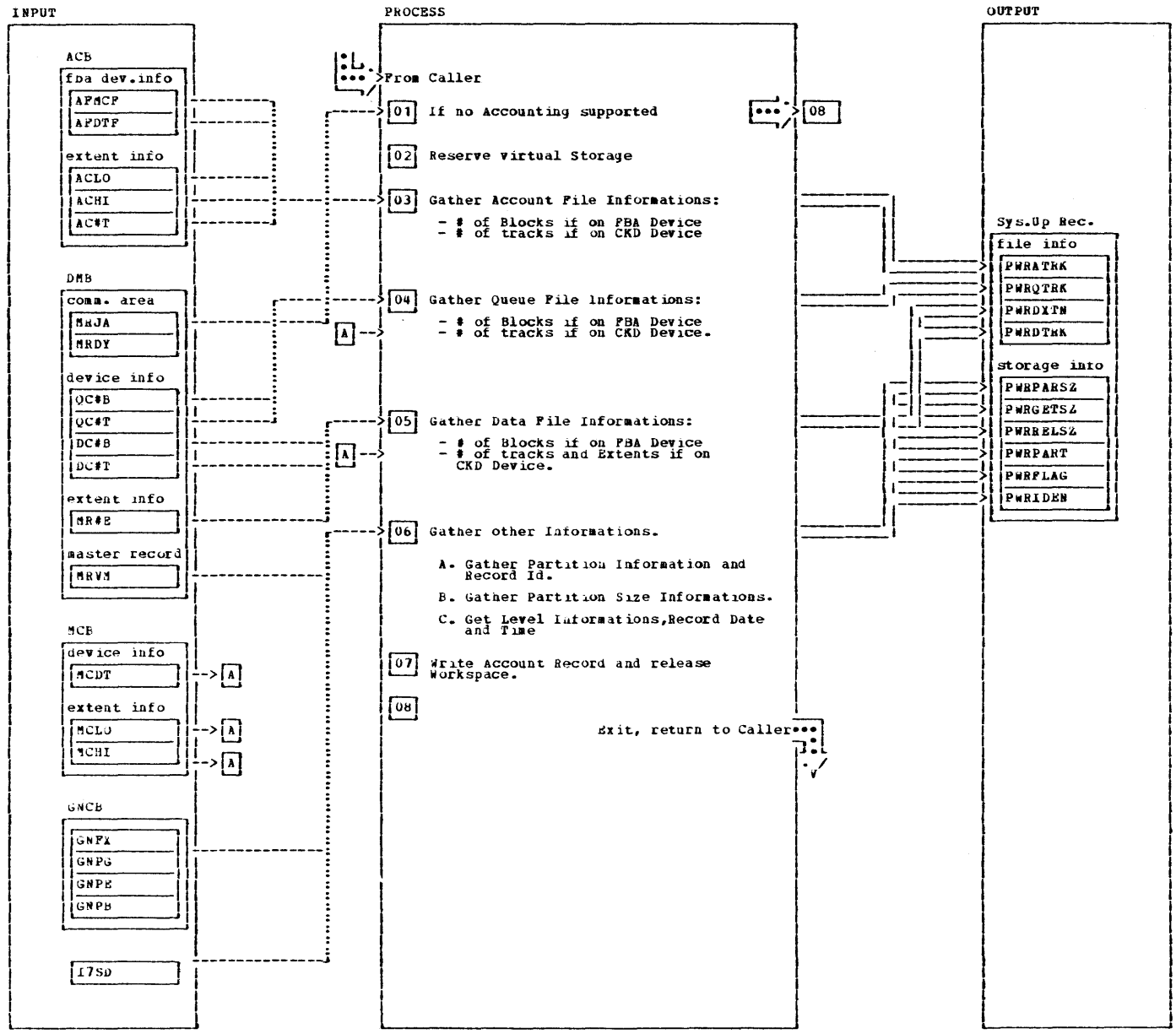
Update master-record and set up Timer task



NOTES	MODULE	LABEL	REF
4 The queue file is unlocked to allow other CPUs to access the queue file	UNLOCK	FN00	$WTQ
If a bad return code (¬=0) is given back, message 1Q85I INTERNAL MACRO CALL FAILED, RC=rrmm is issued and the VSE/POWER initialization is terminated.			$GAM
5 Set up task id, task-save-area address, module-entry-point address			

NOTES	MODULE	LABEL	REF
5 Storage for the Timer task is reserved. If no storage is available message 1Q03I INSUFFICIENT REAL/PFIXED STORAGE ALLOCATED is issued and the VSE/POWER initialization is terminated.		FN24	$RSM
The task control block (TCB) is initialized and the Timer task is attached			$GAM $ATT
6 The command-processor task is set inactive, so that from now on operator commands can be entered			

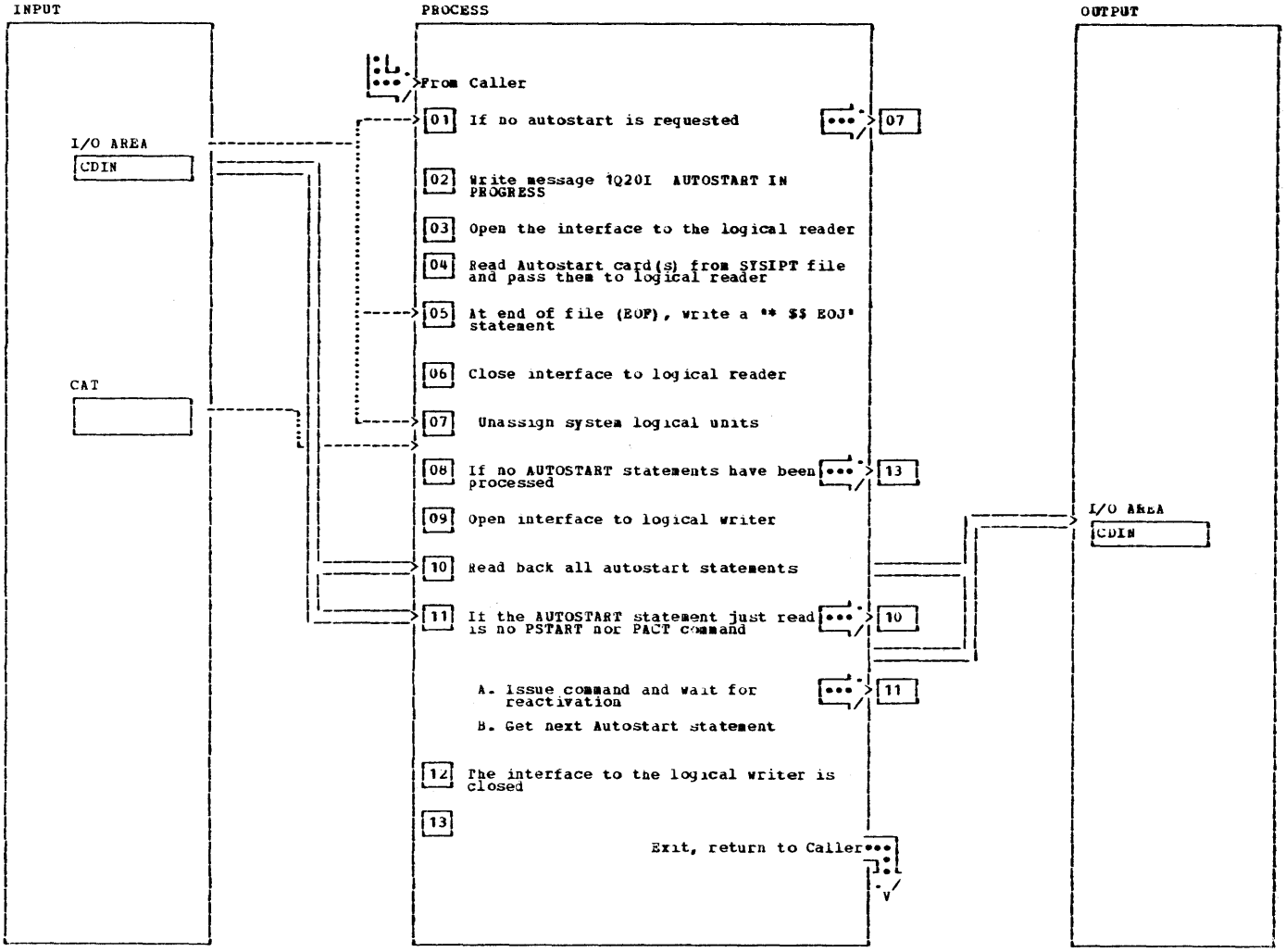
Write System Up Record



NOTES	MODULE	LABEL	REF
2 If no storage available following Msg.will be issued : 1QA6I NO STORAGE AVAILABLE FOR nnnnnn.		I7SAR	$RSV
5 For each defined extent, a MCB is created and to be scanned to accumulate the # of tracks for Data File. The # of Extents-MCB's is hold in MR#E Field in DMB.			
6A Partition Id retrieved from the PIB. Record Id is 0.			

NOTES	MODULE	LABEL	REF
6B <GNFX> * 1024 = Real Part.Size. <GNPG> - <GNPE> = Getvis Size. <GNPG> - <GNPB> +1 = Partition Size.			
6C Level Id retrieved from Section Descriptor. Version/Mod.Level and Record Date from DMB. Time Info via IPW$RDC.			$MDC
7 The Flag which disallows Put Account during Initialization is set off before and set on again after the Macro call.			$PAR
			$MLV

Autostart processing



	NOTES	MODULE	LABEL	REF
1	If the card input area is empty, no (more) autostart processing is required and control is passed to unassign logical units		AS00	
2	Print MESSAGE: 1Q20I AUTOSTART IN PROGRESS			$GAM
3	Interface with the logical reader is set up and an '* $$ JOB ' statement is written onto disk			$OLI
4	All Autostart cards are read from the SYSIPT file and passed to the logical reader by means of the IPW$PLR macro.	GET		$PLR
5	When end of file is occurred, a '* $$ EOJ' statement is written to disk followed by a null record to indicate end of input			$PLR
6	The SYSIPT file is closed and the logical reader interface is closed	CLOSER		$OLI
7	The system logical units are unassigned. - SYSRDR - SYSIPT - SYSPCH - SYSLST		AS12	$ULP
9	The interface to the logical writer is opened in order to read back all AUTOSTART statements			$OLI

	NOTES	MODULE	LABEL	REF
10	All autostart statements are read back			$GLR
11	If the Autostart card is a PSTART or a PACT command a temporary command processor is attached and the PSTART command bypassed to the command processor.		AS20	
11A	Issue command			$WPC
11A	If the autostart statement does not result in the request for reactivation of the command processor (this is the case when a partition is started, R7-=0) a branch is made to read the next autostart statement			
11B	The next autostart statement is read. If any of the statements READER=, PRINTER= or PUNCHES= are found and they contain valid specification, these statements are moved into the command processor control block (CPB). In all other cases a dummy record, indicating invalid record, is moved in the CPB input area. Then the temporary command processor is posted			$GLR
12				$CLI

IPW$$LR - VSE/POWER Logical Reader	
Label	Routine
CC00	Continuation Check Routine
ED00	EOF/EOJ Routine exit
EJ00	End of Job Routine
FQ00	Format Queue Record
INJH	Insert Job Header Record
INDJ-INDH	Insert Data Set Header Record
INJT	Insert Job Trailer Record
JC00	JCL Mode initialization
JE00	JECL Mode intialization
JH00	JECL Statement Routine
LR00	Entry Processing
MC00	JECL or JCL Mode check
MS00	Message Routine
OC00	Command code checking
RD00	3540 physical reader linkage setup processing
UE00	User Exit Routine
WG00	Write Logical Record to disk Routine, get next record

Services Used	
Service	Macro
Task Management	IPW$WFC
Resource Management	IPW$RLR IPW$RSR
Storage Management	IPW$RLW IPW$RSW IPW$RLV IPW$RSV
Message Service	IPW$GAM IPW$RMS IPW$WTO
Timer Service	IPW$RDC
Save/Restore Caller Registers	IPW$SAV IPW$RET

Interfaces used	
Module	Macro
Physical Reader I/O 3540 Physical I/O	IPW$GLR IPW$GLR

Functions used	
Module	Macro
IPW$$AQ	IPW$AQS
IPW$$DQ	IPW$DQS
IPW$$FQ	IPW$FQS
IPW$$PA/PF	IPW$PAR
IPW$$PD	IPW$PDR
IPW$$RQ	IPW$RQS
IPW$$SC	IPW$SRJ

Called by	
Module	Description
IPW$$BR	RJE,BSC Reader Routine
IPW$$ER	3540 Diskette Reader
IPW$$IB	RJE,SNA Inbound Processor
IPW$$I7	VSE/POWER Initialization
IPW$$PR	Physical Reader
IPW$$PS	Print StatusRoutine
IPW$$SA/SF	Save AccountRoutine
IPW$$SM	Spool Manager (X-partition interface)
IPW$$SY	Tape Reader

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

Labels	Chart LR: 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
	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 TO ID is moved from the TCB Remote FROM ID is moved from the TCB		QRTJ(IPW$DQR) QRFJ(IPW$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$DQR)		
	Default origin and target node names are moved from the DMB		QRTN(IPW$DQR) QRON(IPW$DQR)		
	If the current job, allocated on the queue file, is not to be placed in the hold state, branch to..... >	FQ05	LWER(IPW$DTC)		
	Otherwise, set the current job in the hold state.		QRDP(IPW$DQR)		

Labels	Chart LR: 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$RSR
	The job number is updated:				
	The current job number is loaded in register 1 from the DMB.			R1	
	The current job number is stored in the queue record.		QRNO(IPW$DQR) QRJ#(IPW$DQR)		
	The job number is incremented by one and stored in the master record. If >32767 then set to one.			R1	
			MRNO(IPW$DQC)		
	The DMB is released again if not Save Account task.				IPW$RLR
FQ10	If a x-partition request not active, branch..... >	FQ11			
	Move password into queue record.		QRPW(IPW$DQR)		
	Move job number from queue record to SPL.		SPJN(SPL)		
FQ11	Reserve storage for the job header record and anchor it in the TCB.		TCBE(IPW$DTC)		
	Initialize the following default job header record fields:-				
	• job number		NJHGJID		
	• job copy count to one		NJHGJCPY		
	• job flags to "don't recompute pri."		NJHGFLG1		
	• origin node system qualifier to 1		NJHGORGQ		
	• the origin, exec, print, punch node names to the system name		NJHGORGN NJHGXEQN NJHGPRTN NJHGPUNN		
	• job time		NJHGETS		
	If the job is being read from a remote terminal, then set • origin remote, print remote, punch remote, and queue record origin user names to the remid in the form 'Rnnn'		NJHGXEQU NJHGORGR NJHGPRTR NJHGPUNR QROU		
	If it is an SNA remote terminal.... >	FQR4			
	If PRMT LSTROUT and/or PUNROUT are specified for this remid, then convert the value to 'Rnnn' and save it in the job header. Branch..... >	FQ12	NJHGPRTR NJHGPUNR		
FQR4	If SNA is not supported, branch.....>	FQ12			
	If PRMT LSTROUT and/or PUNROUT are specified for this remid, then convert the value to 'Rnnn' and save it in the job header. Branch.....>	FQ12	NJHGPRTR NJHGPUNR		
FQ12	If x-partition request, the PUTSPOOL USERID is moved to the job header record.		NJHGORGR NJHGXEQU NJHGPRTR NJHGPUNR		
FQ14	Register 1 is restored and a return is made to caller via register 4.			R1,R4	

Labels	Chart LR: IPW$$LR - Logical Reader		Modified Data Fields	Reg. Usage	Calls
	Mode Check Routine JECL, Part 1				
	The incoming data record is examined and processed according to the conditions detected.				
MC00	A possible end of data record ID is reset.		TCGP(IPW$DTC)		
	IF JECL EOJ has been read, ignore and get next record. Link to ....>	WG30			
	After return, loop back....>	MC00			
MC02	If the termination switch in the TCB indicates a stop condition, exit is made to task selection.... >	EJ45			
	A possible flush condition is reset.		TCTT(IPW$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 RJE,SNA inbound processor, set input to upper case.		TCCU(IPW$DTC)		
MC05	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			
MC10	A link is made to reserve queue and job header record space....>	FQ00		R4	
	Branch to check the statement in JC mode processing....>	JC00			

Labels	Chart LR: IPW$$LR - Logical Reader		Modified Data Fields	Reg. Usage	Calls
	* * * * $$ JOB Processing * * * * *				
MC20	The JOB operation code ID in the TCB is set and JECL mode is indicated.		LWOC(IPW$DTC) LWPI2(IPW$DTC)		
	The branch index used in the statement scan is set for job name, and a link is made to reserve queue and job header record space..... >	FQ00	LWBI(IPW$DTC)	R4	
	On return, a link is made to handle the JOB statement..... >	JH00		R8	
	Link to update the default job header with the * $$ JOB information, then write the header to the data file .... >	INJH			
	Link to test and insert a possible data set header record..... >	INDJ			
	Link to skip the spooling of the * $$ JOB statement and read next card... >	WG30			
	Branch to handle the next record ...>	JE05			
	* * * * $$ CTL Processing * * * * *				
MC30	The CTL operation code ID in the TCB is set.		LWOC(IPW$DTC)		
	The parameter form switch in the TCB is set to keyword form, and a link is made to handle the CTL statement... >	JH00	LWFS(IPW$DTC)	R8	
	A link is made to skip spooling and to get a new record ..... >	WG30		R4 R4	
	Branch to check what type of input mode is to be applied..... >	MC00			
	* * * * $$ RDR Processing * * * * *				
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. 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		R4	
	Otherwise, reset the data mode switch in the TCB and branch to issue an error message and read the next card..... >	JH0X	LWER(IPW$DTC)		

Labels	Chart LR: IPW$$LR - Logical Reader		Modified Data Fields	Reg. Usage	Calls
	JECL Mode Routine JECL, Part 2				
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..... >	WG05			
	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			
	* * * * // JOB Processing * * * * *				
JE20	If no flush condition is present, a branch is made to treat the current record as data..... >	JE00			
	If no DOS/VSE flush condition present, branch..... >	JE00			
	If the current statement is a /& statement, a branch is made to reset the flush condition..... >	JE25			
	* * * Insert /& for EOJ * * * * * *				
	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$DTC)	R6,R7	
JE25	The flush condition is reset, and a branch is made back to treat the record as data..... >	JE00	TCTT(IPW$DTC)		

Labels	Chart LR: IPW$$LR - Logical Reader		Modified Data Fields	Reg. Usage	Calls
	JECL * * * $$ JOB (Unexpected) * * *				
JE40	This routine handles unexpected JECL JOB statement.				
	Indicate job boundary and unexpected * $$ JOB.		TCJB LWUJ		
	Set return code to mode check			R4	
	Link to set EOD and write the job trailer record..... >	IMJT		R8	
	Branch..... >	WG50			
	JECL * * * $$ CTL * * * * * * *				
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			
	JECL * * * $$ RDR * * * * * * *				
JE60	The return address is set in register 4, which can be used by the next subroutine in case the RDR statement is not allowed in the input stream and a link is made to handle the RDR statement.....>	JH00		R4	
	On return, the return address is set again in register 4 and a branch is made to setup the linkage with the 3540 diskette routine.....>	RD00		R4	
	* DOS/Job Control Mode Routine: Part 1				
	This routine examines the first statement of the job being processed.				
JC00	If the current statement is not a // statement, branch to set default job name..... >	JC20			
JC05	If RJE,SNA inbound processor, set input to upper case.		TCCU(IPW$DTC)		
	If no JCL operation code is found, branch to set default job name..... >	JC20			
	If the JCL operation code is not JOB, branch to set default job name..... >	JC20			
	Otherwise, if no job name is found, branch..... >	JC20			
	This routine is entered if the first statement is a // JOB statement. It sets switches: - jobcard - jobname - DOS job card and links to IPW$$SC.		LWOC(IPW$DTC) LWBI(IPW$DTC) LWPI2(IPW$DTC)	R0,R1	IPW$SRJ
	IPW$$SC validates the jobname and moves it to the queue record field. On return from IPW$$SC, a check is made whether user information is available.				
	If so, sixteen bytes of user information are moved to the queue record space.		QRNI(IPW$DQR)		

Labels	Chart LR: IPW$$LR - Logical Reader		Modified Data Fields	Reg. Usage	Calls
JC20	Link to update the job header record with the information from the // JOB statement and write the job header record to spool..... >	INJH			
	Link to test and insert a possible data set header record..... >	INDJ			
	Reset switches		LWPB		
	Branch to write the // JOB statement	JC30			
	* * DOS/JOB Control, Part 2 * * * *				
JC30	A link is made to write the record to the data file and read the next one >	WG10		R4	
JC35	A link is made to scan the operation code of the next record..... >	OC00		R4	
	On return, the operation code scan routine provides a branch index into a branch table located at the return point, used to branch to the appropriate routine:				
	0: branch to handle data record.. >	JC30			
	4: branch to handle /& statement..... >	WG00			
	8: branch to handle unexpected // JOB statement..... >	JC40			
	12: branch to handle JECL EOJ statement..... >	JC50			
	16: branch to handle JECL JOB statement..... >	JC40			
	20: branch to handle JECL CTL statement..... >	JC60			
	24: branch to handle JECL RDR statement..... >	JC70			
	JCL * * $$ JOB(Unexpected) * * * *				
JC40	Record pointer registers 0 and 1 are set to point to the EOJ record to be inserted. A /& record is spooled directly. Branch to insert *$$EOJ.. >	WG45		R0,R1	IPW$PDR
	JCL * * $$ EOJ * * * * * * * * *				
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 LR: IPW$$LR - Logical Reader		Modified Data Fields	Reg. Usage	Calls
JC55	An unexpected JECL EOJ statement is invalidated by overwriting a $ sign with an asterisk. A branch is made to write the record to the data file..... >	JC30			
	JCL * * $$ CTL * * * * * * * * *				
JC60	The CTL statement detected is handled through a link..... >	JH00		R8	
	On return, a link is made to skip the record and to read the next. ..... >	WG30		R4	
	A branch is made to scan the new record..... >	JC35			
	JCL * * $$ RDR * * * * * * * * *				
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 3540 diskette routine.....>	RD00			
	* * * * * * * * * * * * * * * * * * *				
	Operation Code Check Routine: Part 1				
OC00	This routine scans an input record and checks for JECL and JCL. A 'Normal Record' indication is made in the TCB.				
OC03	If the indicator to insert data set header record is on, then reset and link to do the insertion ..... >	INDH			
	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, indicate JECL mode, and branch to check..... >	OC50	LWP12(IPW$DTC)		
	If neither JCL nor JECL statement, return to caller..... >	(R4)			

Labels	Chart LR: IPW$$LR - Logical Reader		Modified Data Fields	Reg. Usage	Calls
	Part 2				
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 text following // is found, return to caller..... >	(R4)			
	If RJE,SNA inbound processor, set input to upper case.		TCCU(IPW$DTC)		

Labels	Chart LR: IPW$$LR - Logical Reader		Modified Data Fields	Reg. Usage	Calls
0C15	If a // EXEC statement, indicate data break..... >	0C20			
	If a // JOB statement, return to caller..... >	(R4)+8			
	Otherwise, return to the caller.... >	(R4)			
0C20	A data break is indicated, and control is returned to the caller.. >	(R4)			
0C50	If no JECL operation code is found, return..... >	(R4)			
	*JECL Mode * * * * * * * * * * * * * *				
	If RJE,SNA inbound processor, set input to upper case.		TCCU(IPW$DTC)		
0C55	If JECL EOJ statement, return..... >	(R4)+12			
	If statement is an unexpected JECL JOB statement before EOJ statement, return..... >	(R4)+16			
	If the current job has a flush condition, return..... >	(R4)			
	If the current statement is a CTL statement, branch to process it.... >	0C60			
	If the statement is a RDR statement, return to check it..... >	(R4)+24			
	Otherwise, return to write and get the next record..... >	R4			
0C60	The CTL operation code ID is set in the TCB, the form switch is set to 'Keyword', return to caller..... >	(R4)+20	LWOC(IPW$DTC) LWFS(IPW$DTC)		

Labels	Chart LR: IPW$$LR - Logical Reader		Modified Data Fields	Reg. Usage	Calls
	3540 Physical Reader Linkage Setup:				
RD00	If the 3540 diskette is in data mode then indicate that a data set header record must be inserted.		LWER TCF3		
	The interface to the physical diskette reader (IPW$$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. The new physical save area is initialized as follows:				IPW$RSW
	• 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	
	• The 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$DTC)		
	A branch is made to get the next record from the physical 3540 diskette device..... >	WG30			
	End of Job Routine				
EJ00	Job termination time is obtained through an IPW$RDC call.				IPW$RDC
	Job termination time is stored in the queue record. According to conditions detected, a branch is made to the appropriate routine to continue EOJ processing: If no flush condition, branch to add queue record to queue file..... >	EJ30	QRET(IPW$DQR)		
	Otherwise, the flush condition is reset. If VSE job flush condition was detected branch to add queue... >	EJ30	TCTT(IPW$DTC)		
	If unit exception flush of POWER job then branch..... >	EJ10	QRCN		
	Otherwise set cancel code to normal flush condition		QRCN		

Labels	Chart LR: IPW$$LR - Logical Reader		Modified Data Fields	Reg. Usage	Calls
EJ10	Write an account record.				IPW$PAR
	The queue record is deleted from the class queue using IPW$DQS call. The queue record is added to free set using an IPW$FQS call.				IPW$DQS IPW$FQS
	A branch is made to indicate complete queue set. ..... >	EJ40			
EJ30	Set the cancel code to normal.		QNCN(IPW$DQR)		IPW$AQS
	The queue record is added to the queue file using an IPW$AQS call.				IPW$AQS
	The account record address is loaded into register 1.			R1	
	The account record length (58 bytes) is loaded in register 0.			R0	
	An IPW$PAR call is issued to add the account record to the account file.				IPW$PAR
EJ40	The function trace indicator in the TCB is set to 'Complete EOJ'.		TCFT(IPW$DTC)		
	If dynamic diskette process..... >	EJ42			
	If not primary diskette process or if primary diskette process without a high level pws, branch to continue..>	EJ45			
EJ42	Unassgn pub request is set up. The workspace for the low level pws will be released and the high level pws addressed in the TCB. In case of dynamic diskette process, the pointer in the TCB will be set to zero.		TC3W		IPW$ULP IPW$RLW
EJ44	If 'Stop at EOJ' was not indicated, branch to continue..... >	EJ50			
EJ45	If it is an rje task..... >	EJ47			
	Otherwise check if double buffering if not ..... >	EJ47			
	Address first buffer and if active. > Otherwise the active one.	EJ46			

Labels	Chart LR: IPW$$LR - Logical Reader		Modified Data Fields	Reg. Usage	Calls
EJ46	Wait for completion, move selection field and go to task selection.				IPW$WFC
EJ47	Register 9 is set up as task terminator base register, and a branch is made to the task terminator..... >	(R9)+16		R9	
EJ50	A possible end of data record ID is reset.		TCGP(IPW$DTC)		
	If next record not available, branch to..... >	ED05		R7	
	If this is the start of an unexpected new job, reset unexpected job indicator and return to caller		LWUJ(IPW$DTC)	R4	
	Otherwise, check for user exit routine by branching to..... >	UE00			
	EOF/EOJ Exit				
	This routine is entered in case of unit exception. On unexpected EOF (EOF before EOJ) the data block is written to the data file to make release or work space possible, work space is then released and control is passed back to the physical reader. On interrupt by the applicable reader device, the physical reader will pass control back and work space is reserved again.				
ED00	On unexpected EOF, EOB is indicated, and a data block is written.		TCGP(IPW$DTC)		IPW$PDR
ED05	If the reader task has been started with a 3540 diskette reader (altern. diskette process, LWER=X'01' + X'02') there is a dormant linkage with a card reader, and a branch is made to reset the linkage..... >	ED09	LWER(IPW$DTC)		
	If the reader task has not been started for a 3540 diskette reader only, (primary diskette process, LWER=X'08' + X'02'), branch to check for dynamic diskette process..... >	ED07			
	Otherwise check if a high level pws is present. If not, branch to release the LDA..... >	ED10			
	Setup 'unassgn pub' request, release the workspace for the low level pws, address the high level pws in TCB and branch to continue.....>	ED09	TC3W		IPW$ULP IPW$RLW
ED07	If it is not dynamic diskette process (LWER=X'80' + X'02'), branch..... >	ED10			

Labels	Chart LR: IPW$$LR - Logical Reader		Modified Data Fields	Reg. Usage	Calls
ED09	If 3540 in data mode, then indicate insert data set header record.		TCF3(IPW$DTC)		
	The 3540 communication switch is reset.		LWER(IPW$DTC)		
	Register 1 is loaded with the address of the physical 3540 save area address.			R1	
	The physical 3540 save area is released.				IPW$RLW
	The physical card reader save area is chained back to the logical save area.		SVSV(IPW$DTC)		
	A possible end of data record ID is reset.		TCGP(IPW$DTC)		
	Indicate that a data set header record must be inserted.		TCF3		
	A branch is made to continue reading from the card reader..... >	WG42			
ED10	Register 3 is saved as it will be destroyed by the following release work space function.		SVR3(IPW$DSV)		
	Register 1 is loaded with the LDA address.			R1	
	The data LDA is released through an IPW$RLW call.				IPW$RLW
	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$RLW
	Queue space pointers in the TCB are set to zero.		TCQA(IPW$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
	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 LR: 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$RSW
	Real and virtual addresses of the buffer space are stored.		TCDA(IPW$DTC)		
	Register 3, whose contents have been destroyed by the IPW$RSW call, is restored.			R3	
	A branch is made to check the new data record..... >	WG50			
	User Exit Routine				
UE00	The current record is examined and passed to the user exit routine according to the following conditions:				
	• If no user exit is available, return to the caller..... >	(R4)			
	• If the task is the initialization task, return to caller..... >	(R4)			
	• If flush condition, return to the caller..... >	(R4)			
	• If the current record has been user inserted, return..... >	(R4)			
	• If neither JECL nor JCL statement, return to the caller..... >	(R4)			
UE10	If RJE,SNA inbound processor, set input to upper case.		TCCU(IPW$DTC)		
UE15	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/VSE job... >	UE50			
	16: Branch to flush VSE/POWER job. >	UE60			
	20: Branch to PNET Modify record.. >	UE25			

Labels	Chart LR: IPW$$LR - Logical Reader		Modified Data Fields	Reg. Usage	Calls
UE25	Ignore return code since it is valid only for the PNET receiver user exit routine ..... >	(R4)			
UE30	The current record is deleted by bypassing the write to data file:				
	Delete condition is reset for next record.		USCC(IPW$DTC)		
	A branch is made to get the next record..... >	WG40			
UE40	Record pointer registers 0 and 1 are restored from the TCB.			R0,R1	
	Current record pointers registers 0 and 7 are saved in the TCB.		USCC(IPW$DTC)		
	Inserted record pointers are loaded into current record pointer registers 0 and 7.			R0,R7	
	Control is returned to the caller.. >	(R4)			
UE50	If at job boundary, branch to ignore flush..... >	UE80			
	Otherwise, set the VSE/POWER cancel code in the queue record to X'0C' to indicate that only the records of the current DOS/VSE job are to be flushed		QRCN(IPW$DQR)		
	Branch to continue..... >	UE70			
UE60	If at job boundary, branch to ignore flush..... >	UE80			
	Otherwise, set the VSE/POWER cancel code in the queue record to X'60' to indicate that all the records of the current VSE/POWER job are to be flushed.		QRCN(IPW$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$GAM
	Control is returned to the caller.			R4	
	JECL Statement Handler				
	This routine provides the interface with the parameter scan function. Parameter registers are set up prior to IPW$SRJ call.				
JH00	If the operation code is not RDR, branch to scan the statement directly..... >	JH06			

Labels	Chart LR: IPW$$LR - Logical Reader		Modified Data Fields	Reg. Usage	Calls
	If no primary 3540-PWS is present (LWER=X'08'), branch to.....>	JH02			
	Verify wether a higher level PWS is already present and if so, to issue msg 1Q90I and set flush indication. Branch .....>	JH0X			
	Acquire storage for 2.PWS, init and chain it to top. Branch for continuation.....>	JH04	TC3W		IPW$RSW
JH02	If no alternate 3540-PWS is present (LWER=X'01'), branch to.....>	JH03			
	Otherwise check if the statement has been read from the diskette (LWER=02) and if so branch to issue error msg and set flush indication.....>	JH0X			
	to continue branch to.....>	JH04			
JH03	Assume dynamic assignment has to be done. At first check if statement has been read from the diskette (LWER=02) and if so branch to issue error msg and set flush indication.....>	JH0X			
	If there is a PWS already available (TC3W=¬0) use that one and branch to continue .....>	JH04			
	Acquire storage for dynamic PWS, save it's address in TCB, and indicat the dynamic assigned PWS (LWER=X'80') to continue branch to.....>	JH04	TC3W LWER		IPW$RSW
JH0X	A link is made to the message sub-routine to issue message 1Q90I.....>	MS05			
	A flush condition is set in the TCB Branch to flush the current Job.....>	WG30	TCTT		

Labels	Chart LR: IPW$$LR - Logical Reader		Modified Data Fields	Reg. Usage	Calls
JH04	Load the address of the physical work space in register 2.			R2	
	Blank the file name.		PEFI(IPW$DPW)		
	Load the address of the master record in register 3.			R3	
	Set following defaults into physical work space:				
	• Clear option byte (FEED)		PEOP(IPW$DPW)		
	If the option FEED in master record is not on, branch to..... >	JH05			
	• Turn option FEED on		PEOP(IPW$DPW)		
	• Number of diskettes to one		PEND(IPW$DPW)		
	• Volume sequence check option (to NO character blank)		PESC(IPW$DPW)		
	• Verify option (to NO character blank)		PEVE(IPW$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)		
JH06	If no blank characters directly after the JECL op. code, print message 1Q371, and branch..... >	JH09			

Labels	Chart LR: 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			
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	LWP1(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
	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 1037I..... >	MS00		R0	

Labels	Chart LR: IPW$$LR - Logical Reader		Modified Data Fields	Reg. Usage	Calls
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			
	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			

Labels	Chart LR: IPW$$LR - Logical Reader		Modified Data Fields	Reg. Usage	Calls
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)	LWPB(IPW$DTC)		
	Continuation Check				
	This routine is entered when a continuation punch is found in the current record. The current record is written to the data file and the next record is requested from the physical reader. This record, the continuation statement, is syntax checked and an error, if found, is flagged with error message 1Q37I. If the 'last parameter' switch is on, and the continuation statement itself contains a continuation punch, the continuation routine invokes itself.				
	Otherwise, the first parameter on the continuation statement is searched. This parameter should start on one of columns 6 through 16 inclusively. A possible error return to the caller is made to an address four bytes past the normal return address.				
CC00	A normal record is indicated in the TCB. If current statement is not CTL, branch to get next record..... >	CC05	TCBP(IPW$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			

Labels	Chart LR: IPW$$LR - Logical Reader		Modified Data Fields	Reg. Usage	Calls
CC20	If RJE,SNA inbound processor, set input to upper case.		TCCU(IPW$DTC)		
	If the next record is not a JECL statement, branch to issue error message 1Q37I..... >	CC30			
	If the last parameter switch is ON, branch to check continuation column..... >	CC50			
	If column 5 is not blank, branch to issue error message 1Q37I..... >	CC30			
	If parameter starts on any of the columns 6 through 16, inclusively, return to caller..... >	(R3)			
	Otherwise, error pointer register 0 is made to point to column 16.			R0	
	Branch to issue error message 1Q37I..... >	CC40			
CC30	Error pointer register 0 is made to point to column 1.			R0	
CC40	A link is made to message routine..... >	MSG0			
	If no continuation punch is present, error return..... >	(R3)+4			
	Otherwise, branch to check continuation line..... >	CC00			
CC50	If no continuation punch, normal return to caller..... >	(R3)			
	Otherwise, branch to check next continuation line..... >	CC00			
	Subroutine to insert Job Header Control Record (JHR)				
INJH	Set pointer to JHR work area		TC3E	R1	
	Update the JHR fields that may have been modified by the JOB statement.				
	• job class		NJHGJCLS		
	• job name		NJHGJNAM		
	• job password		NJHGPASS		
	• job priority in JES2 format		NJHGPRIO		
	• job disposition		NJHPDISP		
	• job sysid		NJHPSYID		
	• job user information		NJHPUSER		
	Write the JHR to the data file				IPW$PDR
	Release the work area				IPW$RLV
	Return to the caller.....>	0(R8)		R8	

Labels	Chart LR: IPW$$LR - Logical Reader		Modified Data Fields	Reg. Usage	Calls
	Subroutine to insert a Job Trailer Control Record				

INJT	Indicate EOD record		TCGP		
	Reserve work space, if no space is				IPW$RSV
	available, branch..... >	EJ45			
	Initialize the job trailer control		NJTLEN		
	fields		NJTGLEN		
			NJTGTYPE		
			NJTGMOD		
	Write the job trailer to data file				IPW$PDR
	Release the work space				IPW$RLV
	Return to caller.....>	0(R8)		R8	
	Subroutine to insert Data Set Header Control Record (DSH)				

INDJ	Entry when reading the first				
	* $$ JOB statement				
	If the input is not an IBM 5425 MFCU				
	or the input is not from a remote				
	terminal, return to caller			R8	
	If the record length is equal to 80				
	then return to the caller.....>	0(R8)		R8	
INDH	Entry when reading the first IBM				
	3540 diskette record				
	Reserve work space, and if none is				IPW$RSV
	available, branch..... >	EJ45			
	Initialize DSH control fields		NDHCLEN		
			NDHCTYPE		
			NDHCMOD		
	Set the maximum record length equal		NDHCLLEN		
	to present reocrd length.				
	If the input is not from a local 3540				
	then branch..... >	IND8			
	Create within the data set header a		NDHPLEN		
	VSE/POWER subsystem section(short		NDHPTYPE		
	form).		NDHPMOD		
	Indicate the 3540 cuu		NDHPCUU		
IND8	Write the data set header record to				
	the data file				IPW$PDR
	Release work space and return to the				IPW$RLV
	caller.			R8	
	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.				

Labels	Chart LR: IPW$$LR - Logical Reader		Modified Data Fields	Reg. Usage	Calls
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	
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
	The message is printed.				IPW$GAM
	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	

Labels	Chart LR: IPW$$LR - Logical Reader		Modified Data Fields	Reg. Usage	Calls
	An IPW$RMS call is issued to print the statement in error at the remote terminal.				IPW$RMS
	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
MS20	Control is returned to the caller.				IPW$RET
	Write Record, Get Next				
WG00	Set record information in record request word (rec length + address).		TCCC(IPW$DTC)	R6,R7	
WG02	Add record defined in record request word to the data file.				IPW$PDR
WG05	Job boundary is indicated in the TCB.		TCJB(IPW$DTC)		
	Link to set EOD in the TCB and write the job trailer record..... >	INJT		R8	
	Routine return register is set to mode check routine entry point.			R4	
	Branch to skip spooling to data file	WG30			
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$DTC)		
	Data record length is stored in the RRW.		TCRL(IPW$DTC)		
	If not a spool manager request..... >	WG22			
	If not spool queue display request branch..... >	WG22			
	If control record, branch..... >	WG22			
	Otherwise indicate data record.		TCGP		
WG22	If save account/task, set indicator for 'card move.'		TCGP		
WG25	Data record is added to data file through a IPW$PDR call.				IPW$PDR
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			

Labels	Chart LR: IPW$$LR - Logical Reader		Modified Data Fields	Reg. Usage	Calls
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
	Record address is saved in register 6.				
	Record length is saved in register 7.			R7	
	Branch to..... >	WG50			
WG50	If stop condition present, branch to detach..... >	EJ45			
WG53	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			
	If reading from a 3540 data file, branch to write the record and get the next one..... >	WG10	LWER(IPW$DTC)		
	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			

IPW$$LU - VSE/POWER Update LUB/PUB Tables	
Label	Routine
LU00	Function Entry
LU20	Unassign LUB
LU30	Locate unowned PUB
LU40	Locate and assign free LUB
LU50	Release related LUB's
LU60	Release specific PUB
LU70	Locate PUB of assigned LUB
LU80	Locate PUB of programmer logical unit
LU85	Assign/unassign SYSLST to/from given device
LU90	Function Exit
LUP1	Setup 3800 Printer Subroutine

Services Used	
Service	Macro
Storage management	IPW$RLW IPW$RSW IPW$RLV IPW$RSV

Function(s) used	
Module	Macro
IPW$$AS	IPW$IAS

Called byIPW$ULP	
Module	Description
IPW$$CM	VSE/POWER Command Processor
IPW$$I7	VSE/POWER Initialization
IPW$$LM	RJE,BSC Line manager
IPW$$LD	PNET Driver
IPW$$OE	Open 3540 Diskette Routine
IPW$$OF	Poffload task
IPW$$OT	Open Tape Routine
IPW$$PL	Physical List
IPW$$PS	Print Status
IPW$$TR	Task Terminator
IPW$$SA/SF	Save Account
IPW$$XJ	Execution JECL scanner
IPW$$XR	Execution Reader

Labels	Chart LU: IPW$$LU - Update LUB and PUB Tables	Modified Data Fields	Reg. Usage	Calls
	The first 16 bytes constitute the section descriptor: 'LUCS release' 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 SYSLIST to printer 36: unassign SYSLIST		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: VSE/POWER permanent area address	IPW$DPA	R10	
	11: FCB address	IPW$DTC	R11	
	13: Task save area address	IPW$DSV	R13	
	14: caller's return address		R14	
	15: Function base address		R15	
	Function Entry			
L000	Caller's registers are saved. Because the subsequent functions will (or may) change system I/O tables, any concurrent I/O handling activity may produce unpredictable results. Therefore, the system is 'seized' (monopolized) for the duration of function execution through an SVC 22. Parameter register 0 will contain 255 (X'FF') to signal that interrupts are allowed.			IPW$SAV

Labels	Chart LU: IPW$$LU - Update LUB and PUB Tables		Modified Data Fields	Reg. Usage	Calls
	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	
	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 instead 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	

Labels	Chart LU: IPW$$LU - Update LUB and PUB Tables		Modified Data Fields	Reg. Usage	Calls
LU22	Using register 8 as a work register, the logical unit number concerned is retrieved from the CCB.			R8	
	Register 7 is updated to point to the LUB entry concerned.				
	If LUB currently unassigned.....>	LU28			
	Otherwise the PUB pointer is loaded into register 8 and the assignment is cleared.				
	If programmer logical unit.....>	LU26			
	If disk.....>	LU23			
	If not diskette.....>	LU26			
LU23	Point to SYSIN DIB.				
	If SYSIN DIB.....>	LU24			
	Point to SYSPCH DIB.				
	If SYSPCH DIB.....>	LU24			
	Point to SYSLST DIB.				
	If not SYSLST DIB.....>	LU26			
LU24	Clear DIB current address and DIB end address if not SYSRDR or SYSIN.				
LU26	Get pubownership table address.				
	If we do not own this PUB.....>	LU28			
	Release ownership.				
	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 LU: IPW$$LU - Update LUB and PUB Tables		Modified 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 LU3I.			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 LU: IPW$$LU - Update LUB and PUB Tables		Modified 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)		
	Locate and Assign LUB Routine:				
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 LU: IPW$$LU - Update LUB and PUB Tables		Modified 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 LUB's 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 LU: IPW$$LU - Update LUB and PUB Tables		Modified 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/PICL 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 LU: IPW$$LU - Update LUB and PUB Tables		Modified 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 LU: IPW$$LU - Update LUB and PUB Tables		Modified 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	

Labels	Chart LU: IPW$$LU - Update LUB and PUB Tables		Modified Data Fields	Reg. Usage	Calls
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. If LUB concerned is programmer LUB, branch.....>	LU78	SVR2(IPW$DSV)	R6 R8	
	Otherwise, using register 9 as a work register, the number of system LUBs is stored in the first halfword of task register 2 save area, and subtracted from the programmer LUB pointer in register 8 to have register 8 contain the LUB index for the first system LUB.		SVR2(IPW$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 LU: IPW$$LU - Update LUB and PUB Tables		Modified Data Fields	Reg. Usage	Calls
	Identify PUB 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	
	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.				
	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.....>	LU32			
	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.				
	Assign/Unassign SYSLST				
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.....>	LU32		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.				

Labels	Chart LU: IPW$$LU - Update LUB and PUB Tables		Modified Data Fields	Reg. Usage	Calls
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	
	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 to LUB 14 return address				
LUP1	Check whether the TCB belongs to the command processor task. If so, return to caller via register 14. 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	
	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.			R3	

Labels	Chart LU: IPW$$LU - Update LUB and PUB Tables	Modified Data Fields	Reg. Usage	Calls
	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$RSW
	Set up register 4 as a base to the work space area.		R4	
	Copy the storage descriptor into the work space area.	LUWS(LUWS)		
	Save registers 14 and 15.	LUGR(LUWS)		
	The model SETPRT parameter list and the LUB address are moved into the work space.	LUSP(LUWS)	R1	
	Storage is reserved for the service request block (SRB).			IPW$RSW
	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$DSR)	R2	
	Temporarily assign a new register save area.	LUSV(LUWS)	R13	
LUP3	Pass the SETPRT request to asynchronous service for processing IPW$IAS TYPE=SERVICE.			IPW$IAS
	Unassign the temporary register save area and reassign the original.		R13	
	Restore registers 14 and 15. Release the service request block.		R14,R16	IPW$RLW
	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 caller via register 14.			

IPW$$LW - VSE/POWER Logical Writer	
Label	**Routine**
PJ00	Pass Output to physical Routine
PJ88	Put data record Subroutine
PJ98	Get data record subroutine
AC00	Accounting Routine
NJ00	Start of new Job (Queue entry)
EQ30	End of Queue entry
AB00	Abnormal Condition Handler
SU01	Setup Handler
RS01	Restart Handler
SE00	Separator Handler

Services Used	
Service	**Macro**
Task Management	IPW$WFQ IPW$WFO
Resource Management	IPW$RLR IPW$RSR
Storage Management	IPW$RLW IPW$RSW IPW$RLV IPW$RSV
Message Service	IPW$GAM
Disk / Tape Service	IPW$RDD IPW$RDQ IPW$CTT
Timer Service	IPW$RDC

Interfaces used	
Module	Macro
Physical Routine	IPW$PLR

Functions used	
Module	Macro
IPW$$GD	IPW$GDR
IPW$$NQ	IPW$GQS
IPW$$DQ	IPW$DQS
IPW$$FQ	IPW$FQS
IPW$$PA/PF	IPW$PAR

Called by	
Module	Description
IPW$$I7	VSE/POWER Initialization
IPW$$PL	Physical List
IPW$$PP	Physical Punch
IPW$$SA/SF	Save Account
IPW$$SM	Spool Manager (X-partition interface)
IPW$$BW	RJE,BSC List/Punch Routine
IPW$$OB	RJE,SNA outbound Processor

Labels	Chart LW: IPW$$LW - Logical Writer		Modified Data Fields	Reg. Usage	Calls
LWSD	The first 16 bytes constitute the section descriptor: 'LWCS release' On entry, the following register contents are relevant: 10: Permanent area address 11: Address of writer TCB 13: Address of writer task save area. Entry is made at this point whenever any writer task (physical list or physical punch) issues its first IPW$GLR call for a data record. Pass Job Output:		IPW$DPA IPW$DTC IPW$TSV	R10 R11 R13	
LW00	Register 9 is loaded with the routine origin address to serve as the base register. Branch over main routin ..>	NJ00		R9	
PJ00	Get next data record .....>	PJ98			
	If immediate stop is indicated in TCB of task, branch to.....>	EQ72			
PJ10	Reset restart page count.		QRRR(IPW$DQR)		
	If no normal condition is posted in the TCB, a branch is made to..... >	AB00			
PJ11	Is first time for this copy? If no, branch to..... >	PJ15			
	If spool management request, branch to..... >	PJ15			
	A link is made to the separator routine..... >	SE00		R3,R15	
	Set first time indicator off.		LWFT(IPW$DTC)		
	If separator pages/cards are written, start over again; branch to..... >	PJ00			
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	

Labels	Chart LW: IPW$$LW - Logical Writer		Modified Data Fields	Reg. Usage	Calls
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..... >	S040			
	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 not, branch.....>	PJ22			
	Set first time indicator on.		LWFT(IPW$DTC)		
PJ22	If this is not the last copy, then branch..... >	PJ25			
	Read first-in-set queue record, unless this is a tape task or the first in-set queue record is already in storage.				IPW$RDQ
PJ23	If a spool management request, branch to..... >	PJ25			
	A link is made to print end separator pages..... >	SE00		R3,R15	
PJ25	The general purpose byte is reset.		TCGP(IPW$DTC)		
	If the current queue record is the first in set, or if it is tape spooling, a branch is made to skip the read of the first in set .....>	PJ30			
	Otherwise, the first-in-set pointer is moved from queue record to TCB.		TCQW(IPW$DTC)		
	An IPW$RDQ call is issued to read the first-in-set queue record.				IPW$RDQ

Labels	Chart LW: IPW$$LW - Logical Writer		Modified Data Fields	Reg. Usage	Calls
PJ30	The number of processed copies is incremented by one.		LAWS(LADS)	R15	
	If present task is an active spool management (GETSPOOL) task, branch to end-of-job processing..... >	EQ30			
	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	LWNC(IPW$DTC) QRNC(IPW$DQR)		
PJ40	Otherwise, processing of the remaining copies is prepared:				
	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)		

Labels	Chart LW: IPW$$LW - Logical Writer		Modified Data Fields	Reg. Usage	Calls
PJ42	If this is not a 3800 printer, branch to..... >	PJ44			
	Otherwise, address 3800 TCB extension area and increase copy group index by one.		PTE3CGI (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$DTC)		
	A link is made to the physical routine..... >	PJ90		R8	
PJ44	If punch output is being processed..>	PJ00			
	Otherwise a skip to channel one record is passed to the physical routine.....>	PJ90	TCCC(IPW$DTC) TCRL(IPW$DTC)		
	On return a branch is made to handle the current job output.....>	PJ00			
	Put data record subroutine:				
	This subroutine passes each record to the physical routine. Two entry points exists: PJ88 - to indicate mount forms requested for RJE tasks PJ90 - to pass all records to the physical routine				
PJ88	Indicate set up to RJE.		TCCC(IPW$DTC)		
PJ90	Parameter register 0 is set up with the data record address, register 1 is loaded with the record length.			R0 R1	
	If task is in stop state.....>	EQ70			
PJ96	control is passed to the physical writer.				IPW$PLR
	Is request for spool management (GETSPOOL)? If not, return to caller..... >	R8			
	Request for open logical writer interface?				
	If yes, reset indicator and branch to reopen logical writer..... >	NJ00	TCSS(IPW$DTC)		
PJ97	Is request for close logical writer?				
	If yes, reset indicator and branch to close logical writer..... >	PJ25	TCSS(IPW$DTC)		
	Otherwise return to caller..... >	R8			

Labels	Chart LW: IPW$$LW - Logical Writer		Modified Data Fields	Reg. Usage	Calls
	Get data record subroutine:				
PJ98	If no workspace was prior acquired..>	PJ99			
	Otherwise release workspace.		TCF3(IPW$DTC)	R1	IPW$RLV
PJ99	If no skip to channel record was inserted, branch to.....>	PJA0			
	Otherwise, setup skip to channel 1 record and reset current line count and return to caller.....>	R8	TCCC(IPW$DTC) TCRL(IPW$DTC) LWICLCT(LADS)		
PJA0	Get next data record. Truncate record to maximum of 512 bytes if neccessary.			R1	IPW$GDR
	If record is an internal control record (op-code=x'FF'), branch to...>	PJ9A			
	If op-code is x'00', but no list output is being processed, return to caller.....>	R8			
	If first time, turn off switch and branch.....>	PJA4	LWIF1FT(LADS)		
PJA2	If page overflow or no line counting is wanted, return to caller.....>	R8			
	Otherwise increase current line count and compare against maximum. If not higher, return to caller.....>	R8	LWICLCT(LADS)	R15	
PJA4	Setup skip to channel one record and save original record length and return to caller.....>	R8	TCCC(IPW$DTC) TCRL(IPW$DTC) LWISVRL(LADS)		
PJ9A	If not Job header record, branch....>	PJA6			
	Save line count per page and branch.>	PJA0	LWILNCT(LADS)		
PJA6	If not Job trailer record.....>	PJ9B			
	If not end of data is indicated.....>	PJA0			
	Otherwise build NOP-record and return to caller.....>	R8	TCCC(IPW$DTC) TCRV(IPW$DTC)	R14	
PJ9B	Turn off TRC processing flag and indicate first time through now		PTR3RQB(IPW$DTC) LWIF1FT(LADS)		
	If record is a Dataset header record, branch to.....	PJ9D			
	If the printer is a 3800, the SETPRT parameterlist, which is contained in the old FF-record, is passed as a FD-record to the caller.....>	R8	TCCC(IPW$DTC) TCRV(IPW$DTC) TCRL(IPW$DTC)	R15	
PJ9C	Otherwise a dataset header record, consisting of the general section, is built and passed to the caller. It is created with the following: • length, type and modifier • FCB name • UCS name and options • Forms id			R0,R2	IPW$RSV
	Indicate data set header record created and return to caller .....>	R8	TCF3(IPW$DTC)		

Labels	Chart LW: IPW$$LW - Logical Writer		Modified Data Fields	Reg. Usage	Calls
PJ9D	If dataset header record consists of record characteristics change section branch to	PJA0			
	Otherwise locate VSE/POWER section in dataset header record. If not found.....	PJ9J			
	Copy partition id and SYSID into printer TCB extension area and set page overflow flag, if applicable		PTEPID(IPW$DTE) PTESID(IPW$DTE) LW1F10V(LADS)	R15	
	If the printer is not a 3800.....	PJ9X			
	The SETPRT parameterlist which is part of the VSE/POWER section of the dataset header record, is passed as FD-record to the caller.....	PJ9X	TCCC(IPW$DTC) TCRV(IPW$DTC) TCRL(IPW$DTC)	R15	
PJ9J	If printer is not a 3800.....	PJ9X			
	Locate optional 3800 section in data-set header record. If not found.....	PJ9T			
	A FD-record, containing the SETPRT parameterlist is built and passed to the caller. Following information are extracted from the 3800 data set header record section: • Copy count/groupings • Flash name and count • Burst • character arrangement tables • Flags • Copy modification name Indicate to release workspace, next time the routine is entered.....	PJ9X	TCCC(IPW$DTC) TCRV(IPW$DTC) TCRL(IPW$DTC) TCF3(IPW$DTC)	R1,R2	
PJ9T	Build SETPRT parameter list, requesting hardware default printer setup: • Initialize printer • Forms id • Flash name		TCCC(IPW$DTC) TCRV(IPW$DTC) TCRV(IPW$DTC)		
PJ9X	Return to caller.....	R8			

Labels	Chart LW: 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.			R0	
	If the current task is a punch task or active spool management request, a branch is made to handle punch accounting..... >	AC20			
	Otherwise, if a 'space' is indicated in the general purpose byte, a branch is made to handle line accounting.. >	AC30			
	Then a test is made of the command code byte TCCC in TCB to see whether a skip to channel one has been indicated by the Execution Writer.				
	If yes, handle it and increment page counter..... >	AC02			
	If end of page has not been reached, return to caller.....>	RE			
AC02	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	

Labels	Chart LW: IPW$$LW - Logical Writer		Modified Data Fields	Reg. Usage	Calls
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	
	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.		LAER(LADS)	R1	
	If the current task is a list task and not a spool management (GETSPOOL) task, control is returned to the caller.			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)		
			FCB2(IPW$DTC)	R14	

Labels	Chart LW: IPW$$LW - Logical Writer		Modified Data Fields	Reg. Usage	Calls
	Start of New Job:				
NJ00	The job boundary switch in the TCB is set.		TCJB(IPW$DTC)		
	The first time switch is set.		LWFT(IPW$DTC)		
	To start processing a new job's output, a new queue record is obtained.				IPW$GQS
	If a STOP condition is detected (implying end of tape with tape spooling), branch to detach..... >	EQ72			
	Register 5 is loaded with the queue space address from the TCB.			R5	
	If register 5 is non zero, indicating that a queue record has been obtained, a branch is made to continue..... >	NJ30		R5	
	If the current request is GETSPOOL, then return to physical routine.....>	PJ95			
	If the current task is an RJE task then return to physical routine ....>	EQ85			
	If message 1Q34I has already been issued, branch to..... >	NJ05			
	Otherwise, (logical writer, no queue record available) message 1Q34I is issued.				IPW$GAM
NJ05	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
	On return a branch is made to retry retrieving a queue record..... >	NJ00			

Labels	Chart LW: IPW$$LW - Logical Writer		Modified Data Fields	Reg. Usage	Calls
NJ30	Read storage is reserved for the LDA. If the input medium is a VSE/POWER spool tape, the maximum LDA size is requested.				IPW$RSW
NJ32	The real address of the LDA is stored in the TCB.		TCDA(IPW$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 work for the second data file buffer in the TCB.		TC2DW(IPW$DTC)		
	Storage for the second LDA (logical data area) is reserved.				IPW$RSW
	The virtual and real addresses of the LDA are stored in the appropriate disk request word.		TC2DA(IPW$DTC) TC2DV(IPW$DTC)		
NJ35	Work space is reserved for account handling.				IPW$RSW
	Register 4 is set up to address account work space.			R4	
	The work space address is stored in the TCB.		LWAW(IPW$DTC)		
	The task start time is saved for accounting.		LAST(LADS)		IPW$RDC
	The number of copies is moved from the queue record to the TCB.		LWNC(IPW$DTC)		
	If tape spooling is active, the number of copies is reset to 1.		LWNC(IPW$DTC)		
	The output device type as specified in the queue record is compared with the output device type for the previous job as saved in the TCB. If device types match, or if the current task is an RJE task, a branch is made to continue..... >	NJ38			
	If spool management is active (GETSPOOL) then branch to get data. >	PJ00			
	If the device type is unknown X'FF', or if this is Account output.....>	NJ38			
	Otherwise, message 1Q41I is issued.				IPW$GAM

Labels	Chart LW: IPW$$LW - Logical Writer		Modified Data Fields	Reg. Usage	Calls
NJ38	Get data record.....>	PJ98			
	If this is not a list task..... >	NJ40			
	If this is not a FCB load..... >	NJ40			
	If the task is a local writer task, but the new forms id does not match the one on the printer, the current FCB name is zeroed to force loading of the new FCB in all cases.		PTEFCBN (IPW$DTE)	R1	
	Pass FCB load request .....>	PJ90			
	and get next data record.....>	PJ98			
	If this is a not a 3800 printer, or an RJE task, branch to..... >	NJ42			
	The form-number, flash identification, and paper thread request, as specified in the queue record, are compared with the appropriate values of the previous job. If one of them is different, branch to issue message..... >	NJ49			
	Otherwise, branch to..... >	NJ72			
NJ42	The forms ID as specified in the queue record is compared with the forms ID of the previous job, as saved in the TCB. If both IDs do not match, branch to issue message..... >	NJ44			
	If it is not a punch task, branch to>	NJ80			
	If the PAUSE option was not specified, branch to..... >	NJ80			
NJ44	Otherwise, if the current task is not an RJE task, a branch is made to issue forms change message..... >	NJ50			
	If RJE-BSC task, then branch to queue message..... >	NJ46	TCCU(IPW$DTC)		
	Establish addressability in register 1 for the SNA logical unit control block (LUCB).		TCB1(IPW$DTC)	R1	
	If session allows PDIR record branch to avoid queuing message.... >	NJ80	LUPD(IPW$DLU) LUPS(IPW$DLU)		
NJ46	The remote id is loaded in register 0 issue message 1Q40A ON ttt FORMS ffff NEEDED FOR jobname, jobnumber.			R0	IPW$GAM

Labels	Chart LW: IPW$$LW - Logical Writer		Modified Data Fields	Reg. Usage	Calls
NJ48	A link is made to the physical routine..... >	PJ88			
	The first record is read in again...>	PJ90	TCPR(IPW$DTC)		
	and a branch is made..... >	NJ70			
NJ49	If a clear print is automatically issued at end of job time, branch to..... >	NJ4A		R2	
	Otherwise, save current record control word in the TCB extension area.		PTEGWA (IPW$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$DTC)		
NJ4A	Issue message 1QA5A.				IPW$GAM
	Branch to wait for operator response..... >	NJ54			
NJ50	Issue message 1Q40A.				IPW$GAM
NJ54	An IPW$WFO call is issued to wait for the operator command in response to message 1Q40A or 1QA5A.				IPW$WFO
NJ70	If FLUSH was entered, branch to.... >	NJ72			
	If FLUSH HOLD was entered..... >	NJ72			
	The new forms ID in the TCB is set according to the queue record.		LWFI(IPW$DTC)		
	If RJE task branch to..... >	NJ72		R2	
	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.		PTE3CG1 (IPW$DTE)	R2	
NJ80	A check is made if a PSTOP restart has been done. If not, a branch is made to obtain data..... >	PJ10			
	If PSETUP is given..... >	NJ85			
	Indicate restart, get numbers of copies left and get restart page/ card.		LWNC(IPW$DTC) TQRS(IPW$DTC)		
	If copies left equal zero..... >	NJ85			
	otherwise set up copies left.		LWNC(IPW$DTC)		

Labels	Chart LW: IPW$$LW - Logical Writer		Modified Data Fields	Reg. Usage	Calls
NJ85	Indicate restart for accounting.		LASR(LADS)		
	If it is not a 5425..... >	PJ10			
	otherwise do a secondary feed.		TCCC(IPW$DTC)		
	Insert count.		TCGP(IPW$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$DTC)	R8	
	Check the option byte in the master queue record if a clear print is wanted at the end of job time. If so, issue clear print..... >	PJ90	TCCC(IPWDTC)	R2 R8	
EQ34	A link is made to the physical routine..... >	PJ95			
	If restart is not requested, continue at..... >	EQ35			
	Otherwise:				
	Set remaining copy counter to 1,		LWNC(IPW$DTC)		
	indicate restart at EOJ, and		LWEJ(IPW$DTC)		
	continue with Restart Handler..... >	RS00			
EQ35	Get address of SYSCOM.			R2	ASYSCOM
	(VM hand shaking change - CP close)				
	Test if VM=YES option has been specified for supervisor (bit in SYSCOM). If not, continue with normal processing..... >	EQ35A		R2	
	Branch and link to routine to close file when under VM..... >	EQVM		R8	
EQ35A	Release logical data area. Indicate job boundary.		TCJB(IPW$DTC)		IPW$RLW
	If a second logical data area exists, it is released too after any I/O did complete using this buffer.				IPW$RDD IPW$RLW
	The buffer pointers in the TCB (real and virtual LDA addresses) are set to zero.		TCDA(IPW$DTC)		
	A possible restart indication is set to zero.		QRRR(IPW$DQC)		
	If not PFLUSH branch to..... >	EQ36			
	If QRDP=K branch to..... >	EQ37			

Labels	Chart LW: IPW$$LW - Logical Writer		Modified Data Fields	Reg. Usage	Calls
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$DQR)	R1	
EQ39	The queue record is set to HOLD state		QRDI(IPW$DQR)		
	Set termination to flush.		TCTT(IPW$DTC)		
	Otherwise, the queue record is set in HOLD state and the flush hold condition is set to flush.		QRDI(IPW$DQR)		
EQ40	The queue record is deleted.				IPW$DQS
	If neither PFLUSH nor PHOLD was issued..... >	EQ54			
	Reset flush condition and set VSE/POWER cancel code..... >	IA00			
EQ54	If normal EOJ not present, branch to..... >	IA00			
	Otherwise, set normal EOJ cancel code in the Q record.		TCTT(IPW$DCT) QRCN(IPW$DQR)		
	Initialization of the account record is now entered.				
IA00	If accounting is not supported, branch to skip accounting.....>	EQ58			
	If a RDR queue entry is being accessed, branch to..... >	EQ58			
	If the current task is a punch task, a branch is made to bypass output page counting..... >	IA20			
	Otherwise, register 1 is loaded with the total page count from the account work space.			R1	
	The number of pages as kept in the queue record is subtracted.			R1	
	If the result is positive, a branch is made to continue page count initialization..... >	IA10			
	Otherwise, the extra page count in the queue record is set according to the extra page count in the account work space.		QRNE(IPW$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			

Labels	Chart LW: IPW$$LW - Logical Writer		Modified Data Fields	Reg. Usage	Calls
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$DQR)		
IA20	The number of lines or cards in the queue record (QRLC) is subtracted from register 7 contents to determine the number of extra lines or cards.			R7	
	If the result is positive, a branch is made to continue..... >	IA30			
	Otherwise, the line or card count in register 7 is reset.			R7	
	The extra line or card count in the queue record is copied from the account work space.		QRNA(IPW$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$DQR)		
IA40	The number of copies is loaded from LAWS and saved into the queue record		QRNC(IPW$DQR)		
	Line or card count register 7 is set to zero.			R7	
	The current date is stored in the account part of the queue record.		QRDT(IPW$DQP)		
	EOJ time is obtained, and stored in the queue record.		QRET(IPW$DQR)		IPW$RDC
	The saved start time is obtained and stored in the queue record.		QRST(IPW$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	

Labels	Chart LW: IPW$$LW - Logical Writer		Modified Data Fields	Reg. Usage	Calls
	If it is a punch task a branch is made to..... >	EQ57			
	Otherwise get length of account record.			R0	
EQ57	The account record is written to the account file.				IPW$PAR
EQ58	The queue record is added to the free set.		TCFT(IPW$DTC)		IPW$$FQS
	Complete EOJ is indicated in the TCB.				
	The account work space is released.				IPW$RLW
	Task conditions are now checked and branches are taken:				
	If stop at EOJ condition, branch to process detach..... >	EQ70			
	If non-RJE task, branch to get next queue record..... >	EQ60			
	If RJE,BSC or active spool management (GETSPOOL) task, branch to detach task..... >	EQ80			
EQ60	If normal condition or RJE,SNA task, branch to get next queue record.... >	NJ00			
EQ70	If the current task is an initialization task, branch to return to physical routine..... >	EQ80			
	(VM Handshaking change - CP Close)				
EQ72	Test if VM=YES option has been specified for supervisor (bit in SYSCOM). If not, continue with normal processing..... >	EQ72A		R2	
	Branch and link to routine to close file when under VM..... >	EQVM		R8	
EQ72A	Register 9 is set up as task terminator base register, and a branch is made to the task terminator.			R9	
				R9	
EQ80	The queue record space is now released:				
	If nothing to release, branch to... >	EQ85			
	Queue record space is released.				IPW$RLW
	The queue record pointers in the TCB are set to zero.		TCQV(IPW$DTC)		

Labels	Chart LW: IPW$$LW - Logical Writer		Modified Data Fields	Reg. Usage	Calls
EQ85	If RJE,BSC task, return to BSC writer via IPW$PLR macro instruction				IPW$PLR
	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..... >	PJ96			
	(VM Handshaking change - CP Close):				
EQVM	Use IPW$RSW macro to obtain 32 byte area from Storage Manager for parameter list.			R1	IPW$RSW
	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
	Return to caller.....>	R8		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..... >	PJ11			
	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			
	Indicate end of data by clearing register 0.			R0	
	Empty current buffer .....>	PJ95			
	Set linkage to terminator.....>	EQ70			

Labels	Chart LW: IPW$$LW - Logical Writer		Modified Data Fields	Reg. Usage	Calls
AB10	This routine is entered on a flush condition. Message 1Q39I is issued.				
	Possible PSETUP/PRESTART indicator is reset.		TCRS(IPW$DTC)		
	Job boundary and end of data are set in the TCB.		TCJB(IPW$DTC) TCGT(IPW$DTC)		
	The first time switch is set off.		LWFT(IPW$DTC)		
	If active spool management (GETSPOOL) task, then branch to..... >	PJ11			
	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$GAM call is issued.			R0	IPW$GAM
	On return, a branch is made.....>	PJ11			
AB20	For a local writer task, the message is logged.		TCMW(IPW$DTC)		IPW$GAM
	If the current task is a punch task, a branch is made to continue..... >	PJ11			
	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.		TCCC(IPW$DTC) TCRL(IPW$DTC)	R1	IPW$GAM
	If printer is not a 3800.....>	AB20			
	Otherwise set command code to 'select translate table 0', and a link is made to output routine>	PJ90			
AB30	On return, the command code is set to 'skip to channel one' and a link is made to the physical routine..... >	PJ90		R8	
	On return, the command code is set to 'write', and a branch is made back..>	PJ11	TCCC(IPW$DTC)		

Labels	Chart LW: IPW$$LW - Logical Writer		Modified Data Fields	Reg. Usage	Calls
AB40	If this is a punch task, branch to stop task.....	> AB43			
	If immediate stop is requested.....	> AB50			
	If it is not immediate skip (CH 1)..	> PJ11			
	otherwise.....	> AB50			
AB43	If PSTOP immediate was issued, branch to.....	> AB45			
	If no card movement, branch to.....	> PJ11			
	If 3525 punch device, branch to....	> AB44			
	If not 3525 read punch device, branch to.....	> AB45			
AB44	Create blank card to finish punch.				
AB45	Is it not a 5425.....	> AB50			
	otherwise do secondary feed.				
AB50	Indicate EOD.				
	Link to physical routine to empty buffer.....	> PJ95		R0 R8	
	Branch to terminate task.....	> EQ72			
RS00	If it is not a RJE restart branch to	> RS01			
	Get seek address and branch to.....	> PJ00	TCDW(IPW$DTC)		
	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	SPFCOPYG (SPLIST)	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	
	Using register 2 as a work register, the total page number LAEP is incremented by one.		LAEP(LADS)	R2	

Labels	Chart LW: IPW$$LW - Logical Writer		Modified Data Fields	Reg. Usage	Calls
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....	> PU20			
	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....	> PU20			
	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 the 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....	> PU20			
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....	> SU44			
	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

Labels	Chart LW: 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
	Forward space file to skip tapemark.				IPW$CTT
	Forward space record to position past first record.				IPW$CTT
SU60	Restart/setup indicator is reset.		TCRS(IPW$DTC)		
	Restart current counter is reset.		LARC(LADS)		
	If current task is a spool management (GETSPOOL) task, branch to.....	> SU62			
	Check for RJE. If not, branch to..	> SU65			
SU62	Otherwise, branch to the physical routine.....	> PJ88			
	Upon return from the physical routine, branch to.....	> SU70			
SU65	If it is an 3800 printer, issue message 1QA9A, otherwise issue message 1Q40A.				IPW$GAM
	An IPW$WFO macro is issued to wait for the next operator command.				IPW$WFO
	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.....	> PJ98			IPW$GDR
	Check for PSETUP or RESTART. If so, branch to.....	> PJ10			
	Indicate restart.		TCRS(IPW$DTC)		
	Get number of copies left.		LWNC(IPW$DTC)		
	Get restart card/page.		TCRS+1(IPW$DTC)		
	Get group copy index to be used.		PTE3CGI(IPW$DTE)		
	Exit is taken.....	> PJ10			

Labels	Chart LW: IPW$$LW - Logical Writer		Modified Data Fields	Reg. Usage	Calls
	Restart Handler:				
RS01	If restart is already active branch.>	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..... >	RS02			
	If the current task is a list task, a branch is made to bypass setting of restart current card count..... >	RS04			
RS02	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 LW: 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 obsolete 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. A branch is made to restart printing from the calculated restart value..... >	RS24		R8	
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$DTC)		
	If tape spooling, branch.....>	RS70			
	The restart current count is reset to zero.		LARC(LADS)		
	The DMB is reserved for subsequent reading of the first queue record in set.				IPW$RSR
	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			
	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)		

Labels	Chart LW: IPW$$LW - Logical Writer		Modified Data Fields	Reg. Usage	Calls
RS32	The disk address of the 'first-in-set' queue record is moved to the DMB		QCQW(IPW$DQC)		
	If tape spooling, branch.....>	RS36			
	Read first-in-set queue record				IPW$RDQ
RS36	If the current task is a GETSPOOL task, a branch is made to continue. Otherwise, a check is made if it is a punch task. If not, branch to.... >	RS38 / RS40			
RS38	If it is a RDR queue record..... >	RS39			
	The restart value in register 8 is compared to the maximum value for card restart and a branch is made.. >	RS44			
RS39	The restart value in register 8 is compared to the maximum value of # of cards and a branch is made.... >	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
	A branch is made to continue..... >	RS50			
RS48	The DMB is first released.				IPW$RLR
RS50	If it is a spool management (GETSPOOL) task, branch to..... >	RS54			
	If the current task is not an RJE task, a branch is made to issue the message..... >	RS52			
	Otherwise, the remote ID is loaded in register 0, and an IPW$GAM is issued to send the message.			R0	IPW$GAM
	A branch is made to continue output processing..... >	RS54			
RS52	The message is logged.				IPW$GAM
RS54	Reset RESTART current counter		LARC(LADS)		
	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..... >	PJ10			
	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 LW: 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..... >	RS00			
	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.)	RS06			
RS00	The disk address of the next-in-set queue record is moved to the TCB.		TCQW(IPW$DTC)		
	If tape spooling, branch .....>	RS01			
	The next-in-set queue record is obtained. If the current task is a list task and not a spool management (GETSPOOL) task, a branch is made to continue processing page restart..... >	RS04			IPW$RDQ
RS02	If it is a spool management task and a RDR Queue record, branch to.....>	RS03			
	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..... >	RS00			
	Otherwise the restart current card count is set equal to the card count of this queue record, and a branch is made to get the next queue record ..>	RS00	LARC(LADS)		
RS03	Record count is obtained from QRNR and if restart count in R8 is lower than number of records in R0, a branch is made ..... >	RS00	R0		
	Otherwise the restart current card count is set equal to the card count of this queue record, and a branch is made to get the next queue record ..>	RS00	LARC(LADS)		
RS04	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..... >	RS00			
	Otherwise the restart current count is set equal to the page count for this queue record, and a branch is made to get the next queue record ..>	RS00	LARC+2(LADS)		

Labels	Chart LW: IPW$$LW - Logical Writer		Modified Data Fields	Reg. Usage	Calls
RS66	If the restart current count is not zero, a branch is made to continue.....	>RS68			
	If the current queue record is the first queue record, branch	>RS68			
	If it is tape spooling, branch	>RS68			
	The disk address of the first queue record is copied into the TCB, and the record is read in.		TCQW(IPW$DTC)		IPW$RDQ
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 current card/page.		LARC(LADS)		
	The tape is backspaced to the beginning of the file:				
	• A backspace file call is issued				IPW$CTT
	• A forward space file call to skip tapemark				IPW$CTT
	• And a forward space record to position past first record.				IPW$CTT
	A branch is made to continue.....	>RS74			
RS73	The data pointer is moved from the queue record to the TCB.		TCDW(IPW$DTC)		
	When double data file buffering, the seek address of the first data block is moved from the queue record to the 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 LW: IPW$$LW - Logical Writer		Modified Data Fields	Reg. Usage	Calls
RS80	Otherwise, read next data record.				IPW$GDR
	If end of data, branch .....>	RS50			
	If the current task is a list task and not a spool management (GETSPOOL) task, test for a 'skip' operation ..>	RS84			
	Otherwise, if it is a RDR entry, ...>	RS86			
	The general purpose byte is tested if the punch operation causes a feed. If not, a branch is made back to get the next data record..... >	RS80			
	Otherwise, a branch is made to increment the restart current count..... >	RS86			
RS84	If the associated command code is a 'skip to channel 1 immediate' (start of a new page), a branch to..... >	RS86			
	If the associated command code is a X'FD' (SETPRT REQUEST), the SETPRT parameter list is saved in the TCB extension area.		PTELIST (IPW$DTE)	R1,R14	
	Branch to get next data record..... >	RS80			
RS86	The restart current page card count is incremented by one.			R6	
	A branch is made back to test if restart is complete..... >	RS76			
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)		

Labels	Chart LW: IPW$$LW - Logical Writer		Modified Data Fields	Reg. Usage	Calls
	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. 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'.		PTEGWA(IPW$DTE) TCCC(IPW$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)		
RS98	The restart condition is reset by setting the restart current counter to zero.		LARC(LADS)		
	Reset restart active indicator.		TCG2(IPW$DTC)		
	Exit is made..... >	PJ20			
RS99	If restart is not from zero a branch is made back to..... >	PJ20			
	Otherwise a get data record for the first record will be done after a skip to channel 1 is passed to the physical routine.....>	PJ90	TCCC(IPW$DTC) TCRV(IPW$DTC) TCRL(IPW$DTC)		
	Branch to.....>	PJ10			

Labels	Chart LW: IPW$$LW - Logical Writer		Modified Data Fields	Reg. Usage	Calls
	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$DQS
	Branch to..... >	SE01			
SET0	If no separator pages/cards.....>	SE01			
	Read in header queue record: • Backspace record			R3,R6	IPW$CTT
SE01	Register 6 is loaded with the number of separator pages requested.			R6	
	If no separator pages/cards are wanted, return to 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, LWRV LWSR(IPW$DTC)		
	Branch to set up the printer if applicable..... >	SS01			
SE02	Clear out restart information.		TCRS(IPW$DTC)		
SE03	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.....>	SE05	QRER(IPW$DQR)	R1,R2	
SCT0	Set up the record address and the command code in the TCB. Check for end of job. If so, branch to punch the end separator cards... > Check for a 5425. If so, branch to >	SCEJ SCT5	TCRV(IPW$DTC) TCCC(IPW$DTC)		
	Load a record length of 80 into register 1 and store it in the TCB. Fill the buffer with the attention character X'78'.		TCRL(IPW$DTC) SCDS	R1	
	Check the number of required separator cards in register 6, and if necessary, set it to a minimum of 3.			R6	

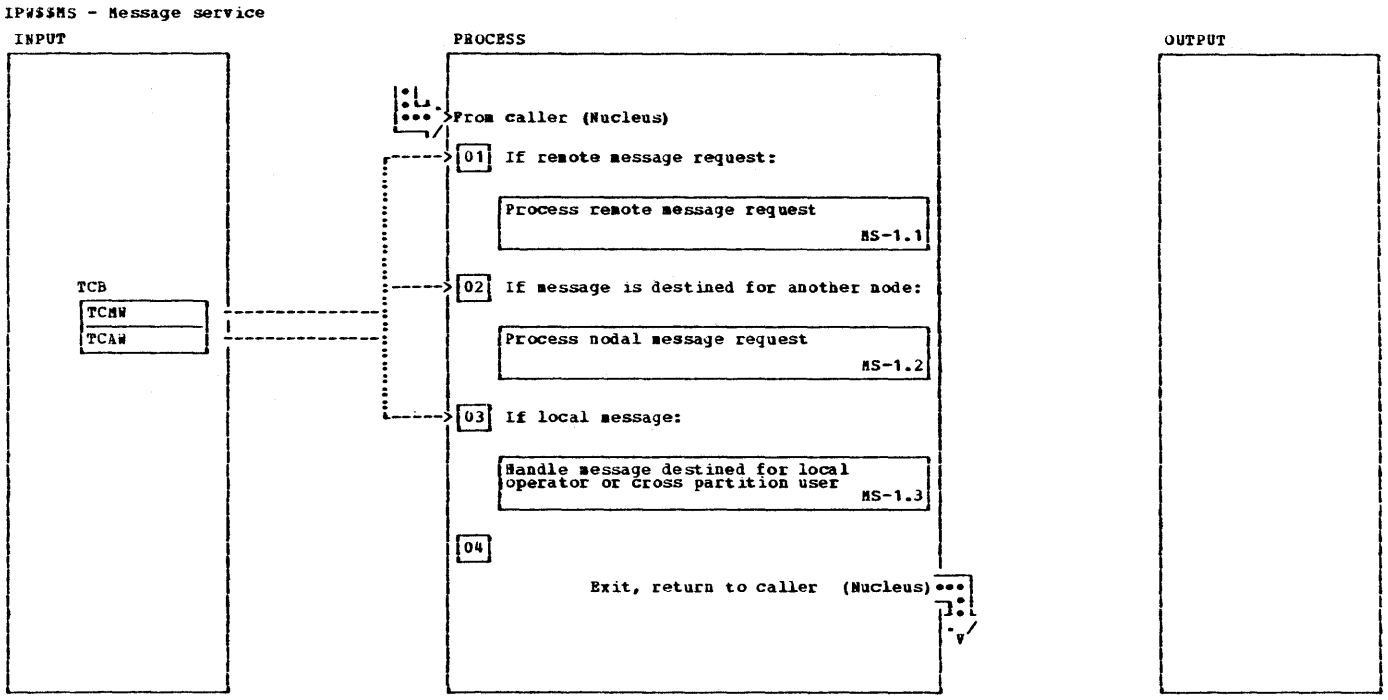
Labels	Chart LW: IPW$$LW - Logical Writer		Modified Data Fields	Reg. Usage	Calls
SCT1	Branch and link to punch the separator cards..... >	SCT8		R2	
	Load the buffer address into register 8.			R8	
	Save the job name.		SCNM(SCDS)		
	Load the character count into register 1, and the starting address into 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$DTC)	R8	
	Check for a stop condition. If so, branch to detach the task..... >	EQ72			
	Set the command code to print and feed (X'41'), and branch and link to the physical routine..... >	PJ90	TCCC(IPW$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	
	When all separator cards have been punched, return to caller via link register 2.			R2	
SE05	The buffer address is loaded in register 8, which serves buffer addressability.			R8	

Labels	Chart LW: IPW$$LW - Logical Writer		Modified Data Fields	Reg. Usage	Calls
	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$DTC)		
	Get start addresses.				
SE0C	Get layer of character until all characters are finished. Print line..... >	PJ90		R8	
	Loop until all 12 lines printed.... >	SE0C			
	Perform "space 3"..... >	PJ90		R8	
	If through with 2nd line, branch to > Get Class and Priority and Number	SE0F	LWJN(LWDS)		
	Translate to displacement. Branch to..... >	SE0A			
SE0F	Add to extra records.		LAER	R1	
SE0G	Add to extra pages.		LAEP	R1	
	If this is last page, branch..... >	SE85			
	Restore request word. The 120-byte buffer is filled with blanks.		TCRW(IPW$DTC)		
	19 asterisks are moved to beginning and end of the buffer. The device address is inserted		LWSL(LWSP) LWS8(LWSP) LWS9(LWSP)		
	The partition id and if applicable the SYSID are moved from the printer TCB extension area into the sep. line		LWS0(LWSP) LWPT(LWSP)		
	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)		

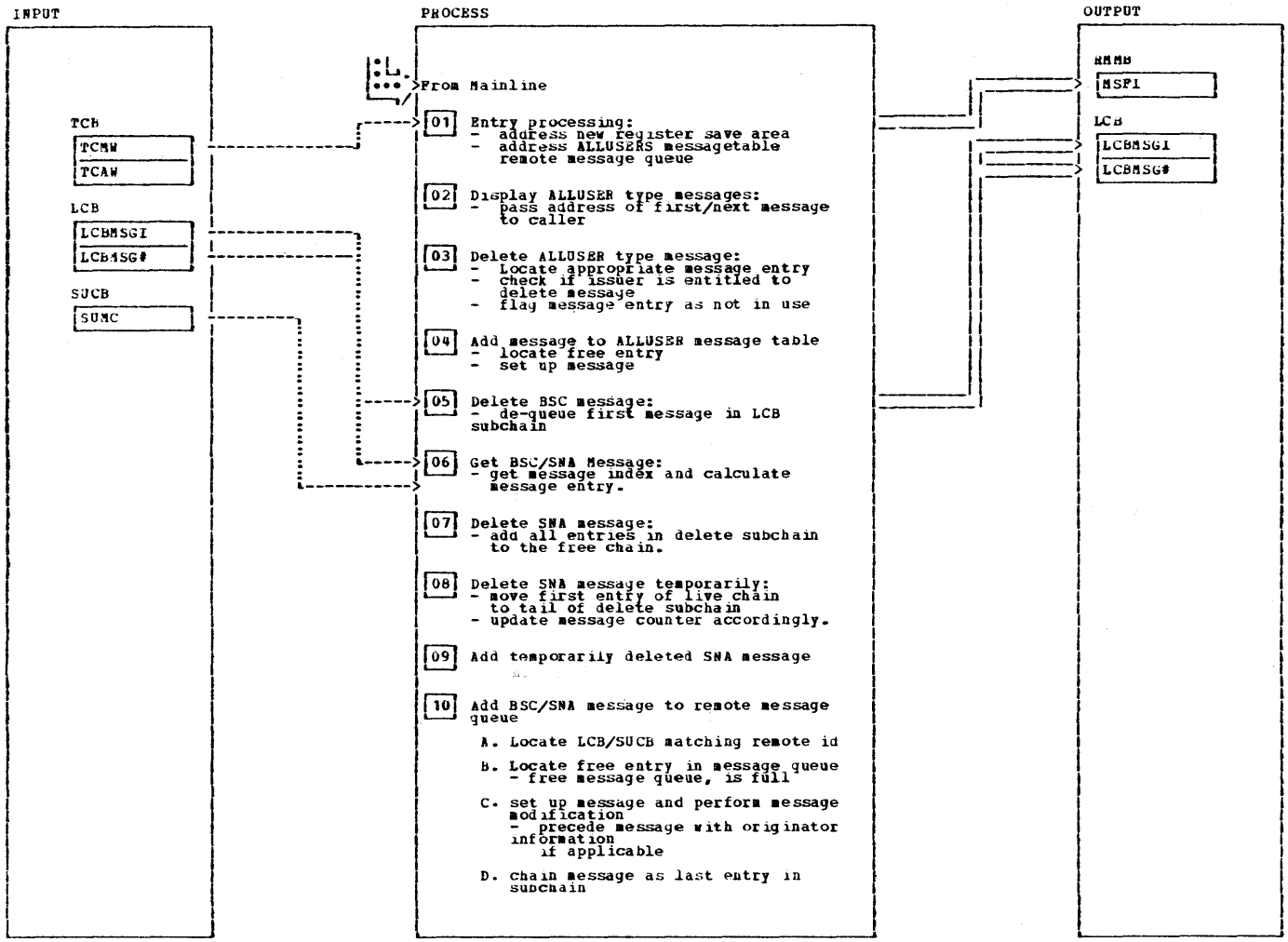
Labels	Chart LW: IPW$$LW - Logical Writer		Modified Data Fields	Reg. Usage	Calls
SE20	Job name is moved from queue record to separator line.		LWS2(LWSP)		
	Job number is moved to separator line.				
	Job number is loaded in register 1 from queue record, converted to packed decimal in work field, and unpacked to separator line.			R1	
			LWCD(LWSP)		
			LWS3(LWSP)		
	Job suffix number is moved to separator line:				
	Job suffix number is loaded in register 1 from queue record, converted to packed decimal in work field, and unpacked to separator line.			R1	
			LWCD(LWSP)		
			LWS4(LWSP)		
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)		
	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 LWMT. R2 is then loaded with the correct month name address.			R2	
	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$RDC call, stored in work field,				IPW$RDC
	and edited into separator line.		LWCD(LWSP)		
	Separator record information is stored in the TCB.		LWS7(LWSP)		
	Separator line length (80) is loaded in register 1 and stored in the TCB.		TCRV(IPW$DTC)		
			TCGP(IPW$DTC)	R1	
SE40	The number of "space 3" to be performed is calculated.				

Labels	Chart LW: 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$DTC)		
	Check for a stop condition. If so, branch to detach the task..... >	EQ72			
	A link is made to the physical routine..... >	PJ90			
	On return, branch back until all lines are spaced..... >	SE60			
	Set the I/O command code in the TCB to 'print line'.		TCCC(IPW$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'.		TCCC(IPW$DTC) TCRL(IPW$DTC)		
	The length is set to one.				
	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			
	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.				

Labels	Chart LW: IPW$$LW - Logical Writer		Modified Data Fields	Reg. Usage	Calls
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.				
	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(SPLIST) SPPCOPYG(SPLIST) SPPFLASH(SPLIST) SPPCPMOD(SPLIST)		
	A link is made to the physical routine..... >	PJ90		R8	
	Return to caller..... >	R2		R2	



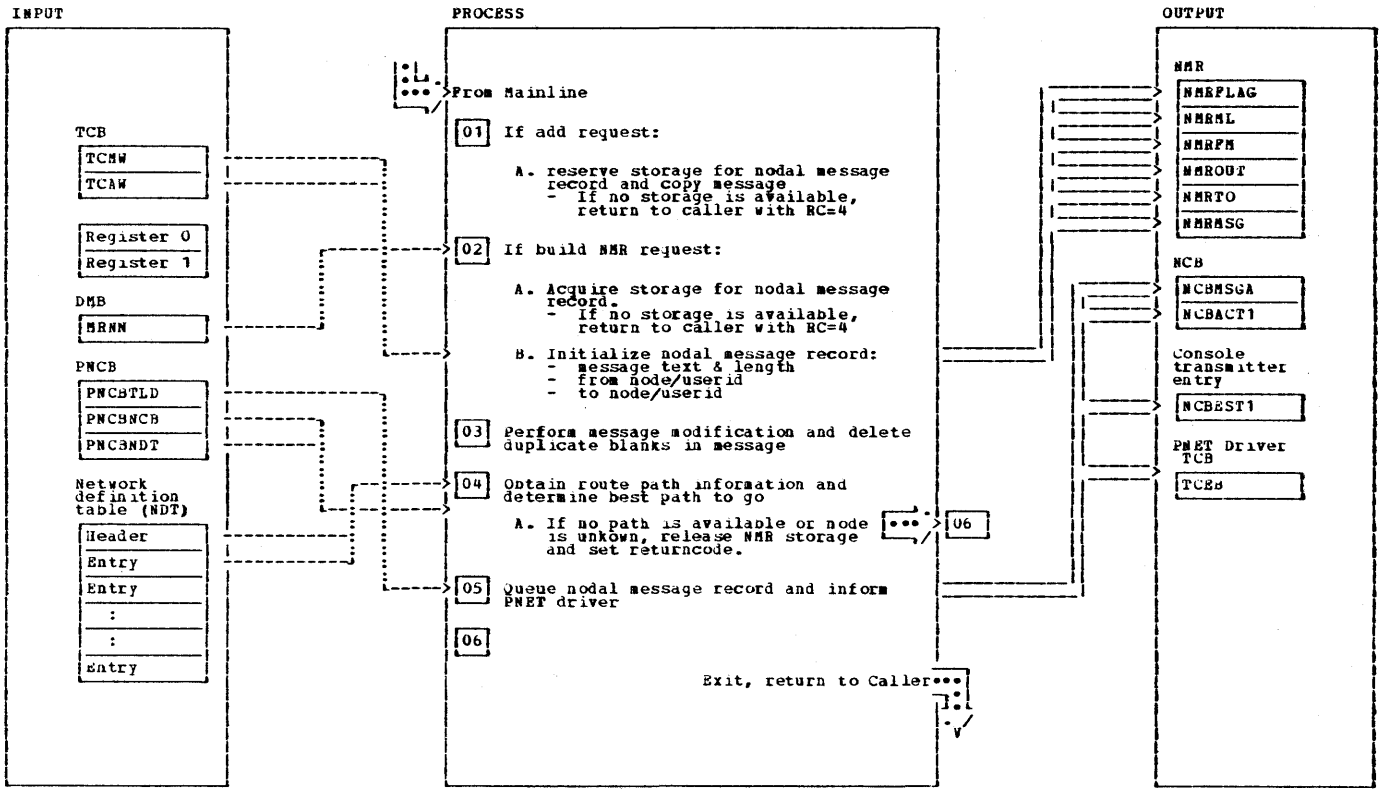
IPW$$MS - Remote message service



	NOTES	MODULE	LABEL	REF
	For a remote message request (IPW$RMS) the function to be performed is indicated in the function indicator byte in the remote message control block. This byte is copied at entry from the first byte of the message control word.			
1	This is the entry point for all remote message requests. The requested function is selected by branching to the appropriate routine.		MS50	
2	If the current entry is the first one (register 1 zero), the ALLUSER message table is addressed. If the current entry is not in use, the next entry is addressed. Otherwise register 1 is setup to point to the message. When all entries have been processed, register 1 is cleared to indicate end-of-message.		DISPROUT	
3	The ALLUSER message number as passed by the caller in register 1 is used to locate the appropriate entry in the ALLUSER messag table. If an invalid message number was specified or the user is trying the delete a message which he is not entitled to, register 1 is set to zero. Note: the central operator can delete any message in the table. The message entry is released (remote id set to x'FF') and return is made to the caller.		DELROUT	
3	If all messages are to be deleted, all entries of the message table are scanned for a matching remote id. If a matching remote id is found, the message entry is flagged as not in use (x'FF') when all entries have been scanned, return is made to the caller.			
4	The ALLUSER message table is scanned to to find an unused entry. If the ALLUSER type message table is full, register 1 is set to zero and return is made to the caller. Otherwise the originator's id is stored in the message table entry. Then the message length is checked and if it is too long, it is set to the maximum (max. 59 bytes).		FFTYPE	

	NOTES	MODULE	LABEL	REF
	The message text and its length are then moved into the entry. If the message being passed is in nodal message record (NMR) format a prefix consisting of originating node name and if present originating user is built is front of the message.			
5	The subchain index (addressing the next message in the LCB subchain) is stored and then updated with the free chain index. Then the free chain index is updated with the message subchain index, and the subchain index which was saved is stored in the LCB. The BSC message counter is decremented by one and stored back.		DELBSC	
6	The current message index is either obtained from the LCB in case of BSC or from the SUCB in case of SNA. If the subchaion is empty(message index = x'FF'), register 1 is set to zero and return is made to the caller. Otherwise the displacement of the message within the remote table is calculated and its address is passed to the caller in register 1.		GETBSC	
7	A check is made to see whether the temporary delete chain is empty. If so, register 1 set zet to zero and return is made to the caller. Otherwise the end of the delete subchain is located and the last entry is et to point to the next free entry. Next the free chain index is updated with the temporary delete chain index and the temporary delete chain index is et to x'FF'.		DELSNA	
8	If the live subchain is emtpt, register 1 is set to zero and return is made to the caller. Otherwise the end of the delete subchain is located and the last entry of the delete chain is updated with the subchain inder. The message entry in the remote message queue is flagged deleted x'FF' and the subchin index in the SUCB is updated with the next message index in the live chain.		DELTEMP	
9	All messages which are currently in the temporarily deleted chain are put back in the live chain in the proper order.		ADDDEL	

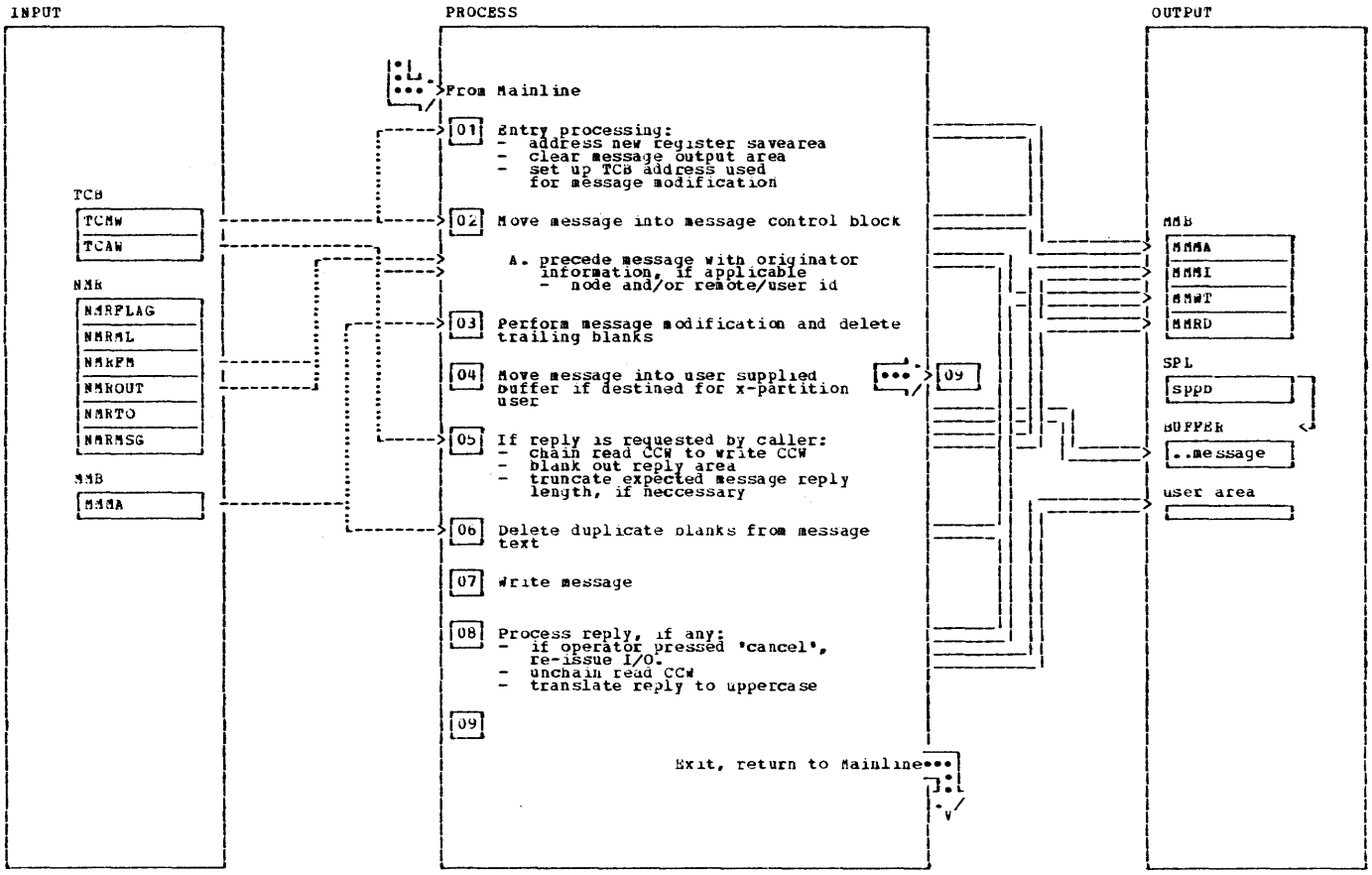
IPW$$MS - Nodal message service



NOTES	MODULE	LABEL	REF
For a nodal message request, which is setup via the IPW$ICS macro instruction, the function code is passed in register 0 while register 1 as well as fields 'TCMW' and 'TCAW' make up the parameter list.			
1 The message or command, which is already in NMR format (nodal message record), addressed by register 1, is queued as last entry in the message/command queue of the appropriate node control block for further transmission.		NMS100	
2 The message contained in the message definition module is addressed and its length is used to obtain enough virtual storage to hold the message in NMR format. If no storage is available, return code 4 is set in register 1 and return is made to the caller. The message to be sent is now copied into the just obtained NMR storage. The originating node id and target node id and if applicable user/remote id are stored in the NMR.			$RSV
3 If the nodal message record contains a message originated from the local system, the message modication routine is called to substitute any variable in the message text. Next the 'blank compression' routine is called to delete two or more consequent blanks in the message. Note: certain messages, identified by the message number, are excluded from the squeeze process.			$GMS

NOTES	MODULE	LABEL	REF
4 Now the NCB chain is scanned to determine the best path to be taken in order to transmit the NMR. - if a connection exists to the prime route, the NMR is queued on this NCB. - If no such connection exists, but a connection exists to the alternate route, if one is specified, the NMR is queued on that NCB. First the network definition table is scanned to get the route1 and route2 information for the target node. The NCB chain is then scanned for a match with the route1 or route2 node name respectively. If a match is found and signon is completed between this and our node, the NMR is queue on this NCB. The NCB chain is locked for the duration of the scan. If the NMR can not be queued on any NCB, the storage used to hold the NMR is released and returncode 12 is set. If the node is unknown, returncode 8 is set.			$RSR
5 The PNET driver is informed that t message/command is put in the queue in order ato attach a console transmitter task, if not already one exists.			

IPW$$MS - Local message service

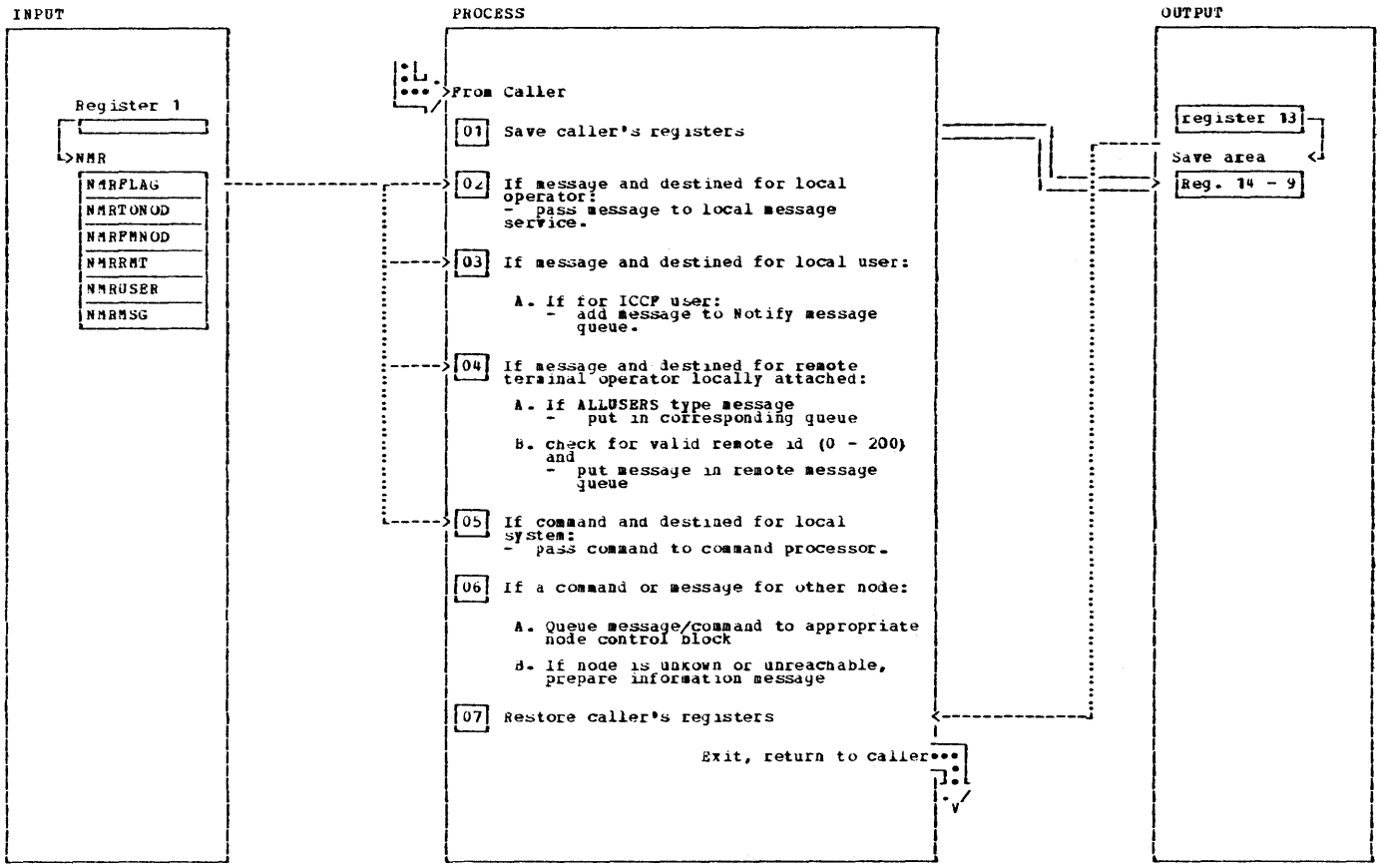


	NOTES	MODULE	LABEL	REF
2	The message address and message length are retrieved from the message request word contained in the TCB of the calling task. The message length is examined for excessive length and truncated, if neccessary before the message is moved into the message output area of the control block. If the message being passed is in nodal message record (NMR) format a prefix consisting of originating node name and if present originating user is built is front of the message. The message reply word is checked for any reason code passed in it. If so, the reasoncode is saved in the save area of register 0 in the message control block.		MS02	
3	The message modification routine is called to substitute any variable which is still in the message text. Next the resulting text is examined for trailing blanks and the length in the write CCW is updated accordingly.			$GMS
4	If the message is destined for a x-partition user, the message which describes best the type of error is returned in the user buffer. In case of a CTLSPOOL request which resulted in a PALTER or PDISPLAY command more than one messages could be issued by VSE/POWER. The last message is returned to the x-partition user.			

	NOTES	MODULE	LABEL	REF
5	The reply address, contained in the TCB of the calling task is now examined to determine whether the caller expects a reply. If so, the expected reply length is retrieved from the user area and truncated if excessive (max. 72 bytes). The read CCW is updated and chained to the preceding write CCW.		MS18	
6	Next the 'blank compression' routine is called to delete two or more consequent blanks in the message. Note: certain messages, identified by the message number, are excluded from the squeeze process.			
7 8	The unit exception bit in the CCB is checked to see whether the operator pressed to cancel key. If so, the message is re-issued. Otherwise the read CCW is unchained from the preceding write CCW. The operators reply is converted to uppercase characters and then moved to the caller's input area.	SVC		$WPC

This page was left blank intentionally.

IPW$$MX - Message/Command Distribution



	NOTES	MODULE	LABEL	REF
	Entry to message/command distribution is made by means of macro IPW$GMS TYPE=DIST. The macro expansion has set up the address of the nodal message record (NMR) in register 1.			
1	The caller's registers are saved in the linkage save area addressed by register 13.			$SAV
2	If the nodal message service represents a message for the local operator (flag = x'00' or x'10'), the NMR is passed to local message service, which in fact places the from node id and if applicable the from user id in front of the message. The message is also destined for the local operator when the user id is 'R000 '			$WTO
3	A message for a local user has a x'20' flag			
3A	The nodal message record is passed to the Notify message queue by issuing the IPW$NFY macro instruction.			$NTY
4	If the message is for a remote terminal operator (flag=x'40') the remote id is taken from the NMR (remote id is always preceded by 'R'). The message is queued by issuing the IPW$RMS macro instruction. On return register 1 contains an indication if the queuing was successful or not.			$RMS
4	Register 1 is zero when either the 'ALLUSERS' queue is full or the remote operator is not logged on.			
4	In the first case a returncode of 4 is set in register 15 if the from node if is the own one. A returncode of 8 is set when the remote operator is not logged on.			
5	Before the 'invoke command processor' routine is called an additional save area is set up.		MD400	$RSW
	If, however the command is in global command syntax the command is discarded.			$ICP
	Next the routine waits for completion of the command processing and releases then the storage used for the additional save area.			$WFC
6	Common services is invoked to queue the message/command to the appropriate node control block. The service scans the network definition table to determine the best path to the other node. Following return codes are given back: - R1 - 0 message/command queued - 4 no storage available - 8 node unknown - c node unreachable		MD600	$ICS
6B	In case of an command, which could not be queued an attempt is made the inform the originator that the target node is either unkown or no connection path exists to reach that node via message IRA2I.			$NTY
7	The caller's registers are restored from the save area addressed by register 13. Following return code in register 15 (RF) are set up: - 0 - normal return - 4 - VSE/POWER RJE not supported - 8 - VSE/POWER PNEF not supported - C - ALLUSERS message queue full - 10 - Remote xxx currently not signed not - 14 - invalid remote id - 18 - no storage to hold message/command - 1C - node unknown - 20 - node unreachable			$RET

IPW$$NQ - VSE/POWER Get Next Queue Set from Chain	
Label	Routine
NQ00	Function Entry
NQ10	Examine Class Table
NQ20	Read Next Record
NQ81	Function Exit

Services Used	
Service	Macro
Resource Management	IPW$RLR IPW$RSR
Storage Management	IPW$RLW IPW$RSW
Message Service	IPW$GAM
Disk / Tape Service	IPW$CTT IPW$RDQ IPW$RDT IPW$WTQ

Called by	
Module	Description
IPW$$LW	Logical Writer
IPW$$NT	Network Transmitter
IPW$$OF	Offload Queues
IPW$$XR	Execution Reader

Labels	Chart NQ: IPW$$NQ - Get Next Queue Set from Chain		Modified Data Fields	Reg. Usage	Calls
NQSD	The first 16 bytes constitute the section descriptor:				
	'NQCS release'				
NQ00	Function Entry				
	Save caller registers 14 through 9 inclusive.		IPW$DSV		
	Reserve queue work area if not present.				IPW$RSW
NQ02	If not tape spooling, branch to.....>	NQ08			
	Skip to next queue set and read the first record into the queue-record work space.				IPW$CTT IPW$RDT
	Load address of message 1Q43I.			R2	
	If end-of-file found, branch to.....>	NQ0b			
	Load address of message 1Q77I.			R2	
	If the queue record is for the current release, branch.....>	NQ03			
	If the queue record is not for a previous release, branch.....>	NQ0b	TCF3(IPW$DTC)		
NQ03	If the contents of record are incorrect, then branch.....>	NQ0b	QRDY(IPW$DQR)		
	The queue identifier is then compared with the task type of the calling task.				
	If they are equal, branch to.....>	NQ85			
NQ0b	Issue message.			R2	IPW$GAM
	Set stop code 'S' in TCB and exit...>	NQ85	TCTT(IPW$DTC)		
NQ08	Set function track indicator to N (get next in progress).		TCFT(IPW$DTC)		
	Set addressability for disk management block (DMB).				
	Lock DMB.				IPW$RSR
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				IPW$RSW
	Reset space address		TCQA(IPW$DTC)		
NQ14	Reset function track indicator (blank)		TCFT(IPW$DTC)		
	Exit.....>	NQ82			

Labels	Chart NQ: IPW$$NQ - Get Next Queue Set from Chain		Modified Data Fields	Reg. Usage	Calls
NQ20	Examine Active Class Chain				
	Translate relative pointer to first-in-class.		QCQW(IPW$DQC)		
NQ30	Read queue record in turn in auxiliary area.				IPW$RDQ
	Spool management GETSPOOL request?				
	If no, branch to.....>	NQ35			
	Positioning requested?				
	If no, branch to.....>	NQ35			
	Positioned on Q-record for job given in spool parameter list (SPL)				
	If no, branch to.....>	NQ40			
	If a job number was specified and it does not match the current queue record, branch to.....>	NQ40			
NQ32	If issuer does not have master authorization, check the password.				
	If a mismatch is found, branch to...>	NQ40			
	Job for Q-record currently printing?				
	If yes, branch to.....>	NQ40			
	Save number of lines and cards.		SPLC(SPL)		
	Q-record dispatchable?				
	If yes, branch to.....>	NQ70			
	Indicate that the record is not positioned.		SPR3(SPL)		
	Reset number of lines and cards.		SPLC(SPL)		
	Disallow GETSPOOL.....>	NQ40			

Labels	Chart NQ: IPW$$NQ - Get Next Queue Set from Chain		Modified Data Fields	Reg. Usage	Calls
NQ35	If the record is not dispatchable, branch to.....>	NQ40			
NQ38	If the record is in execution, branch to.....>	NQ40			
	If POFFLOAD tape function, branch...>	NQ75	TCF2(IPW$DTC)		
	If queue record not in XMT queue....>	NQ39	QCS1(IPW$DQC)		
	If the calling task is a transmitter serving the calling target node, then set queue record to "processing"....>	NQ75	QCTN(IPW$DQC) TCENCB NCBNAME		
	If the target node is known, branch.>	NQP4	NDTNM		
	Set queue record DISP='H', rewrite and branch.....>	NQ40	QCDP(IPW$DQC)		
NQP4	If the transmitter task is serving the node primary route node, branch.>	NQ75	NDTPR		
	Otherwise, reserve the PNCB				IPW$RSR
	If another transmitter is serving the primary route node .....>	NQPA			
NQP8	Release the PNCB. If the transmitter is serving the alternate route node.>	NQ75			IPW$RLR
	Otherwise branch.....>	NQ40			
NQPA	If the primary node transmitter is not signed on.....>	NQP8			
	Otherwise release the PNCB, branch..>	NQ40			IPW$RLR
NQ39	If it is a reader queue record, dispatch it.....>	NQ70			
	If the remote-ids match, dispatch it.....>	NQ65			
NQ40	If all these conditions were not met then the queue entry cannot be processed and the next entry is retrieved.....>	NQ30			
NQ50	If it is not an RJE task nor a GETSPOOL request, the live bit is unposted for the class.		CTQL		
	If the spool files are not shared, examine the next class entry.....>	NQ13			
	Set off the live bit in the correct class entry (RDR,LST,PUN) in the sysid class table for the issuing system's sysid.				
	Examine next class entry.....>	NQ13			
NQ65	If it is an RJE task, branch to.....>	NQ40			
NQ70	If a PNET target shared system was not specified, then branch.....>	NQ75	QRSID(IPW$DQC)		
	If this system is not the target shared system, then branch.....>	NQ40	MRSY(IPW$DQC)		

Labels	Chart NQ: IPW$$NQ - Get Next Queue Set from Chain		Modified Data Fields	Reg. Usage	Calls
NQ75	Set execution switch.		QCXS(IPW$DQC)		
	Indicate executing system.		QCSY(IPW$DQC)		
	Write record back.				IPW$WTQ
	Reserve queue space if necessary and save its real and virtual addresses.		TCQA(IPW$DTC)		IPW$RSW
	Store queue record seek address.		TCQW(IPW$DTC)		
NQ80	Move queue record from auxiliary area to queue space.		QRDS(IPW$DQR)		
	Function Exit				
NQ81	Set the function track byte in the TCB to C'I' to indicate open for input.		TCFT(IPW$DTC)		
NQ82	Check whether the TCB belongs to a command processor task or to 'save account'. If so, branch to.....>	NQ85			
	Unlock the disk management block.				IPW$RLR
NQ85	Check if queue space is present. If so, branch to.....>	NQ89			
	Check if the requesting task is an RJE,SNA writer task. If not, branch to.....>	NQ89			
NQ86	Establish addressability to the LUCB in register 8. Establish addressability to the device entry in the SUCB in register 8.				
	Indicate no output available.		SULIS(IPW$DSU)		
NQ89	Return to calling routine.	R14			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. For FBA devices it is not necessary to convert from relative to absolute, because READ/WRITE will be done with relative block number only. If FBA device is used, store relative block number (R1) in register 1 (relative block number). Return to caller via register 14.			R0,R1, R3 R14	

IPW$$NU - VSE/POWER Nucleus : Services	
Label	Routine
	Task Management
TA01	Task initiation (includes IPW$ATT macro)
TD01	Task initiation (includes IPW$DET macro)
TM01	Task selection (includes ipw$WFx macro)
	Resource Management
RM02	Reserve resource (includes IPW$RSR macro)
RM51	Release resource (includes IPW$RLR macro)
	Storage Management
SM03	Reserve workspace (includes IPW$RSW and IPW$OLI macros)
SM51	Release workspace (includes IPW$RLW and IPW$CLI macros)
	Message Service
MM01	Local message service (includes IPWWTO, IPWWTR and IPW$GAM macros)
MM51	Remote message service (includes IPW$RMS macro)
NM10	Nodal message service (includes IPW$ICS macro)
NS10	Notify message service (includes IPW$NTY macro)
	Disk Service
DM20	(includes IPWRDQ, IPWWTQ, IPW$RDD, and IPW$WTD macros)
	Tape Service
TP20	(includes IPWRDT, IPWWTT, and IPW$CTT macros)
	Timer Service
TR01	GETIME (includes IPW$RDC macro)
TS25	Timer interval service (includes IPW$STM macro)
	Validation Service
VA01	(includes IPW$VDA macro)
	Set Remote mask
SR10	(includes IPW$SRM macro)
	Get trace entry
TZ10	(includes IPW$GTE macro)
	Virtual Storage Management
VS01	Reserve workspace (includes IPW$RSV macro)
VS51	Release workspace (includes IPW$RLV macro)
VS91	Unchain workspace (includes IPW$UNV macro)

Services Used	
Service	Macro
Task Management	IPW$WFC IPW$WFD IPW$WFL
Resource Management	IPW$RSR IPW$RLR
Storage Management	IPW$RLW IPW$RSW
Message Service	IPW$GAM

Labels	Chart NU: IPW$$NU - VSE/POWER Nucleus Task Management	Modified Data Fields	Reg. Usage	Calls
TMSD	The first 16 bytes constitute the section descriptor: 'WCB release'			
	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 NU: IPW$$NU - VSE/POWER 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$DPA)		
TD10	If no more than two tasks in existence (three, when queue/data files shared), check for termination in progress: If so, post terminator ECB.		TCGW(IPW$DTC)		
TD30	Release work space occupied by TCB.				IPW$RLW
	Enter task section.....>	TM02			
	Task Selection				
	Register at entry:				
	12: address of next instruction to be executed when task is next dispatched.			R12	
TM01	Save task registers 12 through 9		TCTR(IPW$DTC)		
TM02	Inclusive unpost VSE/POWER master ECB		PAEB(IPW$DPA)		
	Check if timer interval expired, if yes, branch to timer service routine	TS20		R1,R2	
TM10	Scan the task selection list:				
	11: Addresses TCB			R11	
	12: Addresses selection routine.			R12	
TM20	1. W state: If VSE/POWER master ECB is posted.....>	TM02			
	If running with queue/data file shared, and if no other task is waiting for the DMB/ACB, the nothing-to-do-ECB is posted.		SSNW(IPW$DQC)	R1	
	Otherwise, wait for posting of it (SVC 7) and branch.....>	TM02			
TM30	2. L state: If lockword of the resource zero, enter dispatch routine.....>	TM90		R3	
	If the queue and/or account file shared, and the resource is either the DMB or ACB (if Account file is shared), prevent cancelation of time interval.		SSWK(IPW$DQC)	R1	
	address next TCB.....>	TM10			
TM50	3. M and Q state: Scan control block list for posting of the traffic or event bit. If found, store address of relevant control block in register 1 in task save area and enter dispatch routine.....>	TM90	TCR1(IPW$DTC)	R2	

Labels	Chart NO: IPW$$NO - VSE/POWER Nucleus Task management		Modified Data Fields	Reg.	Calls
TM60	4. R state: Resets the task selection field to R state after redispatching the task when a page fault has been handled.		TCRC(IPW$DTC)		
	Enter the dispatch routine..... >	TM90			
TM80	5. C and S state: If traffic or event bit posted in control block:				
	Test if unrecoverable error has occurred. If not, enter dispatch routine..... >	TM90			
	Test if task accepts I/O error. If yes, branch..... >	TM90			
	If writer task (LST or PUN), then enter dispatch routine..... >	TM90			
	Test for I/O error during data file access. If so, enter dispatch routine..... >	TM90			
	Test for RJE. If line manager, ignore the error..... >	TM90			
	Test for PNET. If Pnet Driver, ignore the error..... >	TM90			
TM82	Set termination indicator in TCB (I/O error occurred).		TCTT(IPW$DTC)		
	Address task-terminator phase (IPW$$TR) and set entry point.			R2	
	Store entry point in register 12 (task selection).		TCRC(IPW$DTC)		
	Enter dispatch routine..... >	TM90			
	Otherwise, (no posting) address next TCB..... >	TM10			
	6. I, O, and P state: address next TCB..... >	TM10			
TM80	7. B state: If the related task is posted on a BSC line event, unpost it and enter dispatch routine..... >	TM90	TCBP(IPW$DTC)		
	Otherwise, address next TCB.... >	TM10			
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	

Labels	Chart NU: IPW$$NU - VSE/POWER Nucleus Common Nucleus Service Entry		Modified Data Fields	Reg. Usage	Calls
COM0	Check if called from outside of nucleus. If not, branch..... >	COM4			
	Save register 9.		TC09(IPW$DTC)		
COM4	Set up in register 9 the second base register of IPW$$NU. Branch to service routine..... >			R9	
EXIT	Check if called from inside of nucleus. If so, return to caller.. >	R2			
	Otherwise restore caller's register 9 and return to caller..... >	R2			
	Registers at entry: 2: return address 3: address of control block to be reserved.			R2 R3	
RM01	Reserve Resource If resource is available, set ownership (address of TCB) in the control block. Branch to exit of nucleus function. >	EXIT			
RM02	Otherwise, save registers 0-1. Check if resource to be locked is DMB; if so branch..... >	RM10	RMSV		
	Check if resource to be reserved is ACB. If not, branch..... >	RM20			
	Check if account file is shared. If not, branch..... >	RM20			
	Address work-to-do ECB using register 1, and branch..... >	RM12		R1	
RM10	Check if queue file is shared. If not, branch..... >	RM20			
	Address work-to-do ECB using register 1.				
RM12	Post work-to-do ECB.			R1	
RM20	Wait for the resource and retry the test..... >	RM01			IPW$WFL
RM51	Release Resource Reset lockword to zero if it is owned by caller. Branch to exit of nucleus function. >	EXIT			

Labels	Chart NU: IPW$$NU - VSE/POWER Nucleus Storage Management		Modified Data Fields	Reg. Usage	Calls
	Reserve Workspace				
	Registers at entry:				
	0: return code			R0	
	1: length of required work space			R1	
	2: return address			R2	
SM01	Lock SCB (explicitly coded).				IPW$WFL
	Save caller registers 14 through 5 inclusive in SCB.		SCTR(IPW$DSC)	R14-R5	
SM02	Set ownership in control block. Round length of required work space up to next 32-byte boundary.		SCLK(IPW$DSC)		
SM10	Address first or next page. If next page pointer not zero..... >	SM40		R3	
SM20	Storage assignment table scan routine:				
	Examine the page control byte:				
	X'00' FFIX this page and format it..... >	SM30			
	X'40'				
	X'80' calculate next page address and retry..... >	SM20			
	X'C0' no space available..... >	SM21			
SM21	Unpost ECB in SCB.		SCEB(IPW$DSC)		
	Examine the return codes (in register 0, byte 3:)			R0	
	X'00' immediate exit..... >	SM26			
	X'04' bypass message and wait..... >	SM24			
	X'08' issue message and wait..... >	SM22			
SM22	Issue warning message to operator 1Q59I WAITING FOR REAL/FFIXED STORAGE				IPW$GAM
	Reset return code to X'04'.		SCR0(IPW$DSC)		
	Update count of tasks waiting for storage.		NRTW(IPW$DPA)		
SM24	Restore caller registers.			R14-R5	
	Save space length.			R3	
	Unlock SCB.		SCLK(IPW$DSC)		
	Wait for posting of ECB in SCB.				IPW$WFC
	Restore space length and retry..... >	SM01		R1	

Labels	Chart NU: IPW$$NU - VSE/POWER Nucleus Storage Management		Modified Data Fields	Reg. Usage	Calls
SM26	Immediate exit (caller will handle wait for ECB):				
	Set address of ECB of SCB in caller register 1		SCR1(IPW$DSC)	R1	
	Restore caller registers.			R14-R5	
	Unlock SCB.		SCLK(IPW$DSC)		
	Return to caller..... >	SM81			
SM30	Page Format Routine				
	Indicate page in use in storage assignment table.		SCTR(IPW$DSC)		
	Store address of new page in previous page.		PCVN(IPW$DPC)		
	Fix the page (SVC 67).			R1	
	Update statistical information in CAT:			R1	
	• Increment number of fixed pages.		NRPC(IPW$DPA)		
	• Calculate maximum of previous field.		NRPM(IPW$DPA)		
SM32	If machine working in virtual mode only, skip to prevent SVC9..... >	SM33			
	Retrieve real address of new page (SVC 69).			R0	
SM33	Store page control words.		PCRA(IPW$DPC)		
	Store first BCW pointer.		PCFA(IPW$DPC)		
	Indicate last page in chain (zero pointer).		PCVN(IPW$DPC)		
	Initialize first-in-page BCW.		PCFB(IPW$DPC)		
	Initialize last-in-page BCW.		PCLB(IPW$DPC)		
	Clear work space in new page to zero.		PCWS(IPW$DPC)		
SM40	Buffer Scan Routine				
	The following conditions may occur:				
	• Buffer in use examine next				
	• Buffer too small buffer..... >	SM42			
	• Scan of page unsuccessful..... >	SM10			
	• Size of required work space matches buffer size..... >	SM45			
	• Buffer size larger than required (split).				
	First buffer equals unused space.				
	Second buffer equals required space.				

Labels	Chart NU: IPW$$NU - VSE/POWER Nucleus Storage Management		Modified Data Fields	Reg. Usage	Calls
Sm45	Update BCWs: • Store appropriate length values. • Set address of owning TCB in BCW for second (or only) buffer unless OWNER=SYS was specified.				
	Reset user registers: • Real address of work space in register 0. • Virtual address of work space in register 1.		SCR0(IPW$DSC) SCR1(IPW$DSC)		
	Exit..... >	SM80			
	Release Work Space Register 1: virtual address of work space.			R1	
Sm51	Check register 1 for zero (no work space to release); if so branch ...>	EXIT			
	Lock SCB. Save caller registers (14 through 5) inclusive.				IPW$RSR
	Post ECB in SCB (indicate storage returned). Update BCWs.		SCEB(IPW$DSC)		
	The following conditions may occur: • Combination unused spaces possible • The page contains no more active buffers.				
	Reset user-registers to zero, and clear work space no longer needed (including BCW no longer needed).		SCR0(IPW$DSC) SCR1(IPW$DSC)		
	Release the page: Update page pointers.				
	Store new page in the previous page.		PCVN(IPW$DPC)		
	Store previous page in next page.		PCVP(IPW$DPC)		
	Free the page (SVC 68).				
	Record page out of use in storage assignment table.				
	Decrement number of fixed pages.		NRPC(IPW$DPA)		
Sm80	Unlock SCB.				IPW$RLR
	Restore registers and exit..... >	EXIT			

Labels	Chart NU: IPW$$NU - VSE/POWER Nucleus Message Serive		Modified Data Fields	Reg. Usage	Calls
	Local Message Service				
MM01	Save the return address.		TCRG(IPW$DTC)	R2	
	Lock the local message control block.				IPW$RSR
	Save registers 14 through 8.		MMSV(IPW$DMM)	R14-R8	
	Load the address of the message module (IPW$$MS) into register 15 and branch and link to perform the message service..... >	24(RF)		R15	
	Upon return from IPW$$MS, restore registers 14 through 8.			R14-R8	
	If the hold operand was specified in the message request word (TCMW), branch to..... >	MM05			
	Unlock the local message control blk.				IPW$RLR
MM05	Restore the return address.			R2	
	Return to exit..... >	EXIT			
	Remote Message Service				
MM51	If RJE is not supported, branch ....>	EXIT			
	Save the return address.		TCRG(IPW$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$RSR
MM53	Lock the remote message control block.			R3	IPW$RSR

Labels	Chart NU: IPW$$NU - VSE/POWER Nucleus Message Service		Modified Data Fields	Reg. Usage	Calls
	Load the address of the message module (IPW$$MS) into register 15 and branch and link to perform the message service..... >	To(RF)		R15	
	Upon return from IPW$$MS, reload register 3 with the address of the message control block, restore registers 14 through 9.			R14-R9 R3	
	Check for SNA. If not, branch to.. >	MM55		R3	
	Check if the request is from the command processor or the SNA manager, If so, branch to..... >	MM55			
	Unlock the SNA control block.				IPW$RLR
MM55	Unlock the remote message control block.				IPW$RLR
	Restore the return address.			R2	
	Exit..... >	EXIT			
	Get message Text Registers at entry: 0: destination (remote ID or zero) 1: message index number 2: return address 3: address of user-supplied message area				
GM10	Save the return address		TCRG(IPW$DTC)		
	Load the address of the message definition module in register 2 and add the length of the standard storage descriptor to it. The message index number is multiplied by four to give a displacement in the message address table.			R2	
	Restore the return address.			R2	
	Check if user-supplied message area; if not, ... >	GM20			
	move message from definition module to user area and exit ..... >	EXIT			
GM20	Check if return of message address in definition module was requested. If not, branch ..... >	GM25			
	Reset flag byte and exit..... >	EXIT	TCMW(IPW$DTC)		
	(Note: Register 1 contains addressed message).				

Labels	Chart Message	NU: IPW$$NU - VSE/POWER Nucleus Service		Modified Data Fields	Reg. Usage	Calls
GM25		Store message address in TCB, Check if message is for central operator.		TCMW(IPW$DTC)		
		If not, branch.....>	GM30			
		Set address of reply area in TCB to zero, to indicate no reply wanted, and branch to.....>	MM01			
GM30		Indicate ADDNRM request and branch to.....>	MM51			
		Nodal Message Service				
NM10		If no PNET support, branch to.....>	EXIT			
		Lock the remote message control block				IPW$RSR
		Save registers 14 to 9		MSSV(IPW$DMS)		
		Branch and link to the message module to perform the required function....>	20(RF)		RE,RF	
		Upon return from $$MS, reload R3 with remote message control block address and restore registers 14 to 9			R3	
		Unlock the remote message control				IPW$RLR
		Restore the caller's return address and exit.....>	EXIT			

Labels	Chart NU: IPW$$NU - VSE/POWER Nucleus Message Service		Modified Data Fields	Reg. Usage	Calls
	Notify Message Service				
NS10	If message is in NMR-format, branch..>	NS90			
	If no target node name is specified, use the LOCAL node name as target node, and branch to.....>	NS20		R1,R3	
	Otherwise set up function to build an NMR and branch to.....>	NM10		R0	
NS20	If no target userid is specified....>	NS80			
	If the userid is not an RJE userid ('Rnnn '), branch to.....>	NS90			
	If RJE is not supported, branch to..>	EXIT			
	Otherwise convert userid into a RJE remote-id and branch to.....>	GM00		R0,R1 R3	
NS80	Prepare to send message to local operator, branch to.....>	GM10			
NS90	If Notify is not supported, branch..>	EXIT			
	Save caller's return address and lock communicator information block.		TCRH(IPW$DTC)		IPW$RSR
	Save Registers 14 to 9		CIBSVC		
	Branch and link to Notify support to add the message to the notify message queue.....>	20(RF)		RF	
	Upon return reload R3 with address of communicator control block address and restore registers 14 to 9			R3	
	Unlock the communicator control block.				IPW$RLR
	Restore the caller's return address and exit.....>	EXIT			

Labels	Chart/Disk	NU: IPW$$NU - VSE/POWER Nucleus Service		Modified Data Fields	Reg. Usage	Calls
		Registers at entry:				
		0: Return address			R0	
		1: Disk request word (DRW)			R1	
DM20		The first byte of DRW is used to locate MCB (via module control block table in CAT).				
DM21		Lock MCB.		MCLK(IPW$DMC)	R3	IPW$RSR
		Save address of DRW.		MC$1(IPW$DMC)		
		Wait on previous I/O completion.				IPW$WFC
		Save disk request word address		MC$T(IPW$DMC)		
		Address CCB and wait for completion of previous I/O.			R1	IPW$WFC
		If error has occurred (I/O error, wrong length), branch to..... >	DM25			
		Otherwise branch to continue..... >	DM30			
DM25		Locate the TCB which issued the I/O in error.				
		If TCB cannot be found, branch to.. >	DM30		R2	
		Set termination indicator (U) in TCB.		TCTT(IPW$DTC)		
		Set proper function track byte (to G or P).		TCFT(IPW$DTC)	R1	
		Address terminator phase (IPW$$TR) and set entry point.				
		Store entry point in register 12 (task selection).			R12	
		Give task ownership of MCB concerned.		MCLK(IPW$DMC)		
		Branch to..... >	DM21			
DM30		If no I/O should be issued (NOP-operation) branch..... >	DM05		R1	
		If I/O request is for C-K-D device, branch..... >	DM42			
		Initialize channel program (FBA)				
		• current block number		MFRD(IPW$DMC)		
		• type of operation (read/write)		MFOB(IPW$DMC)		
		• number of FBA blocks to process		MFRW(IPW$DMC)		
		• move read address of buffer		MF*B(IPW$DMC)		
				MFRW(IPW$DMC)		
		(Note: when the master queue record should be read in/written, the number of FBA blocks to process is taken from the disk-request word.)				
		Branch..... >	DM43			

Labels	Chart Disk	NU: IPW$$NU - VSE/POWER Nucleus Service		Modified Data Fields	Reg. Usage	Calls
DM42		Initialize channel program (CKD):				
		• Move seek address		MCSA(IPW$DMC)	R1	
		• Move command code and data address		MCRW(IPW$DMC)		
		• Move record number to sector		MCSE(IPW$DMC)		
		• Save virtual address of buffer		MCTV(IPW$DMC)		
		• Save TCB address of owner		MC$T(IPW$DMC)		
DM48		Execute channel program (SVC0).			R0	
		If this is not the data file, branch to..... >	DM00			
		If the last request was double buffer request, the buffer is freed:				
		Get buffer address (if any) and release buffer.			1	IPW$RLW
		If this is not a double buffer request, branch to..... >	DM50			
		If writer task (LST or PUN), branch to..... >	DM05			
		If CCB is complete, branch to..... >	DM50		R1	
		Attempt to get 2nd buffer.				IPW$RSW
		If no buffer is available, branch to>	DM50			
		Set up new buffer addresses.		TCDA(IPW$DTC)	R0,R1	
		Indicate buffer cleared (N).		TCDB(IPW$DTC)		
		Save old buffer address.		MCTV(IPWDMC)		
		Branch to..... >	DM05			
DM50		Restore registers.			R0,R1	
					R3	
DM60		Indicate request handled		MC$T(IPW$DMC)		
		Wait for completion.				IPW$WFC
		If wrong length, or unrecoverable I/O error occurred, exit to task selection..... >	TM82			
DM65		Unlock MCB.		MCLK(IPW$DMC)	R3	IPW$RLR
		Task selection is forced to allow any higher priority task waiting for the resource to get it at that point.				IPW$WFD
		Exit..... >	EXIT			

Labels	Chart NU: IPW$$NU - VSE/POWER Nucleus Tape Service		Modified Data Fields	Reg. Usage	Calls
	Registers at entry:				
	0: Return address			R0	
TP20	Retrieve tape control block (TBb) address from TCB.			R2	
	Set address of CCB in register 1.			R1	
	Execute channel program (SVC 0) and wait for completion.				IPW$WPC
	If any error other than wrong length exit to task selection..... >	TM82			
	If neither EOV nor EOF detected, branch to..... >	TP50			
	Examine CCW to indicate EOF for input mode or EOV for output mode.		TBFG(IPW$DTB)		
TP50	Return to caller.			R2	

Labels	Chart NU: IPW$$NU - VSE/POWER 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	
	Move updated time value from partition-communication area to master record		MRDY(IPW$DQC)	R2 R3	
	Exit.....	EXIT			

Labels	Chart NU: IPW$$NU - VSE/POWER Nucleus 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	
	Exit:				
	Return code in register 0				
	R0=0, normal exit				
	R0=4, error 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 partition control block.			R3	
	The CCW address is checked to determine whether it is on a double word boundary and whether it is in the user partition, the SVA, or the LTA (only if user-owned). If the CCW is not a valid address, exit..... >	EXIT			
VA05	The data area address and the length are obtained from the CCW.			R0,R1	
	If the length is higher than 512, exit..... >	EXIT			
	Calculate end address of data area.			R1	
	If the length is zero and it is not a TIC operation, exit..... >	EXIT			
VA07	If it is a read operation with no data transfer, exit..... >	EXIT			
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 user-owned logical transient area (LTA). (Note: The LTA boundaries are established at initialization time.)				

Labels	Chart NU: IPW$$NU - VSE/POWER Nucleus Validation Service		Modified Data Fields	Reg. Usage	Calls
VA12	If the addresses of the user data area are higher than the partition limits, they are matched against the limits of the SVA: If it is a read/sense operation and the addresses are outside of the partition or LTA, return to caller via error exit.			R3 R2	
	If the addresses are invalid, exit with error indication 4..... >	EXIT		R0	
	If the addresses are valid, exit with register zero being zero..... >	EXIT		R0	

Labels	Chart NU: IPW$$NU - VSE/POWER Set Remote Mark in Remote Table		Modified Data Fields	Reg. Usage	Calls
	Registers at entry: 0 - indication (turn on/off) 1 - remote id 2 - return address				
SR10	Copy indication in register 12. Divide the remote id by eight (this has the effect that the byte displacement in the remote table and the remainder are used as an index to address a bit mask.)			R12 R0,R1	
	If bit posting was requested, branch..... >	SR20			
	Otherwise use remainder of division to address bit mask. This bit mask is used to turn off appropriate bit in remote table. Branch to common exit..... >	Exit			
SR20	Use remainder of division to address bit mask. This bit mask is used to turn on appropriate bit in remote table. Branch to common exit..... >	Exit			

Labels	Chart NU: IPW$$NU - VSE/POWER 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$DTC)		
	Modify PSW in partition save area to force entry into VSE/POWER task selection.		PSAD(IPW$DPA)		
	If page fault in progress, zero page fault register request word.			R13	
	Otherwise, save current page fault request and TCB address.				
	Return to supervisor.			R7	
	Page Fault Processor				
PF03	Calculate base address for nucleus code.			R10	
	Post VSE/POWER master ECB.		PABB(IPW$DPA)		
	Post VSE/POWER partition dispatchable.			R0,R1, R15	IREADY
	Address TCB of the task (from save area).			R14	
	Clear page fault request in TCB.		TCPF(IPW$DTC)		
	Set the task dispatchable.		TCSF(IPW$DTC)		
PF04	Scan task selection list for any other outstanding page fault:				
	• Register 13 loaded with page fault request (if any) or zero.			R13	
	• Register 14 loaded with address of TCB (if found) or zero.			R14	
	The page fault request (register 13) and TCB address (register 14) are saved.				
	Return to supervisor..... >	0(R7)		R7	

Labels	Chart NU: IPW$$NU - VSE/POWER Nucleus Attention Interface Appendage		Modified Data Fields	Reg. Usage	Calls
	Registers at entry:				
	0: Length (minus 1) of command code			R0	
	1: Address of command			R1	
	2: Address of end of input buffer			R2	
	14: Normal return address			R14	
	15: Entry point address of routine			R15	
	Upon return register 15 contains:			R15	
	0: VSE/POWER processed the command				
	4: The command does not exist in VSE/POWER				
	8: The VSE/POWER Command Processor is active				
AI00	Save registers 12 through 11 inclusive.			R12-R11	
	Save buffer end address.			R6	
	Calculate base address for nucleus code.			R10	
AI10	If the command code is incorrect:				
	• restore registers			R12-R11	
	• set return code (R15=4) and return to caller..... >	0(R14)			
AI20	Address command processor TCB.			R11	
	If the command processor is active:				
	• Restore registers			R12-R11	
	• Set return code (R15=8) and return to caller..... >	0(R14)			
AI22	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.		CPOP(IPW$DTC)		
			CPCM(IPW$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$DTC)		
	Initialize command processor TCB:				
	• Clear register save area		TCTR(IPW$DTC)		
	• Store address of CCB		TCR7(IPW$DTC)		
	• Set base register		TCR8(IPW$DTC)		
	• Set entry address		TCRC(IPW$DTC)		
	• Set task dispatchable.		TCSF(IPW$DTC)		
	Post VSE/POWER master ECB.		PAEB(IPW$DPA)		
	Post VSE/POWER partition dispatchable.			R0,R1, R15	TREADY
	Restore registers and exit..... >	0(R14)		R12-R11	

Labels	Chart NU: IPW$$NU - VSE/POWER Nucleus RJE, BSC Channel End Appendage		Modified Data Fields	Reg. Usage	Calls
	Registers at entry:				
	1: Address of CCB			R1	
	8: Entry address			R8	
	7: Exit address			R7	
CE00	Check for channel end, device end, or unit check. If not, return to supervisor.....>	4(R7)		R7	
	Save registers 4, 5, 6, and 7.				
	Calculate base address for nucleus code and address line manager TCB.			R7,R8	
	If I/O completed on RJE,BSC line branch.....>	CE40			
	Address NCB and copy address of next CCW from CSW.		NCBLCCW (IPW$DNC)		
	Queue received input buffer to PNET Driver buffer queue.		TCBQ(IPW$DTC)		
	Post PNET driver dispatchable and branch.....>	CE80	TCEB+2(IPW$DTC)		
CE40	Copy address next CCW from CSW. Set the line manager to dispatchable.		TCEB+2(IPW$DTC)		
CE44	Scan the appendage queue for the last CCB in the chain.			R6,R7	
CE48	Store last CCB in chain.				
CE80	Post VSE/POWER master ECB.		PAEB(IPW$DPA)		
	Post VSE/POWER partition dispatchable			R0,R1 R15	IREADY
	Restore registers 4, 5, 6, and 7.			R6,R7	
	Return to supervisor.....>	4(R7)		R7	

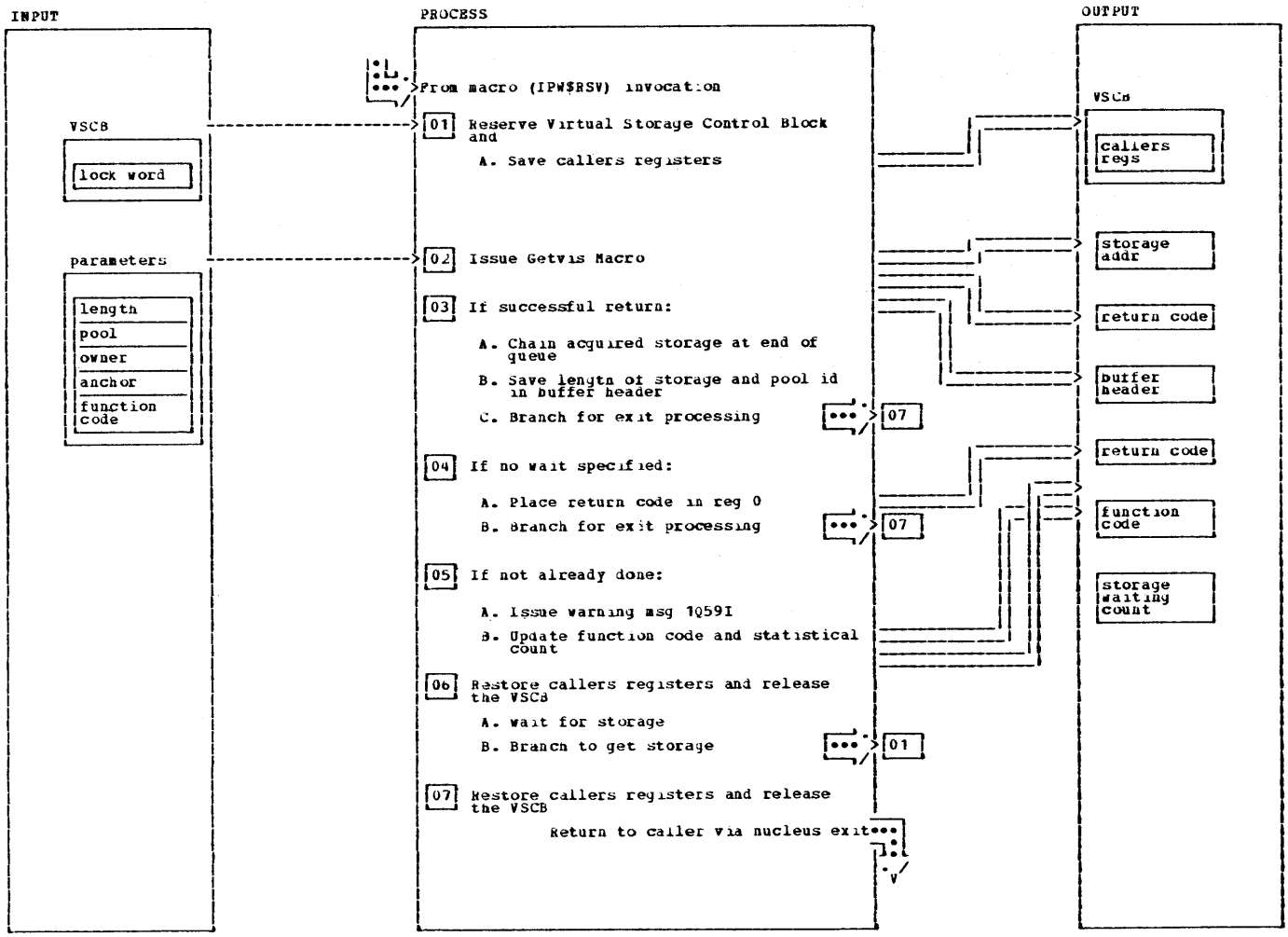
Labels	Chart NU: IPW$$NU - VSE/POWER Nucleus Hot Reader Appendage		Modified Data Fields	Reg. Usage	Calls
	Registers at entry:				
	3: PUB address of the device			R3	
	8: Return address			R8	
	15: Entry address			R15	
HR00	Save registers 9 and 10.			R9-R10	
HR10	Scan reader tasks TCBs for a matching device:				
	If found indicate reader device end occurred		TNG2(IPW$DTC)		
	If found and task is inactive -				
	Set task dispatchable		TCSF(IPW$DTC)		
	Post VSE/POWER master ECB		PAEB(IPW$DPA)		
	Post VSE/POWER partition dispatchable			R0,R1, R15	TREADY
HR30	Restore registers and return to supervisor via register 8.			R9-R10 R8	

Labels	Chart NU: IPW$$NU - VSE/POWER 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(R8)		R10	
SC12	If no interception required..... > (disposition N).	SC11			
SC14	If a previous request pending:				
	Restore register 10, return to supervisor..... >	0(R8)			
	Save register 0 through 3 inclusive.		SCS0	R0-R3	
	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$DTC)	R9	
	Post VSE/POWER master ECB.		PAEB(IPW$DPA)	R0,R1,	TREADY
	Set VSE/POWER partition dispatchable			R15	
	If job account interface is existing scan the SIO table (if present) in the accounting table for matching device and increment counter.			R1,R2, R3	
	Restore registers and return to supervisor via register 8.			R10 R0-R3	

Labels	Chart NU: IPW$$NU - VSE/POWER 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 registers 10 and 11.		R10,R11	
	Calculate base address of nucleus code.		R10	
	Unpost ECB		R0	
	Locate PDB via partition communication region.		R1	
	Store address of parameter list in PDB.	TLCB(IPW$DDE)		
	Obtain requestor's task id from system communication area and store it in PDB.	TLRQ(IPW$DDE)	R11	
	Locate execution reader TCB via PDB.		R1	
	Post its ECB.	TCEB(IPW$DTC)		
	Post VSE/POWER master ECB.	PAEB(IPW$DPA)		
	Post VSE/POWER partition dispatchable			THREADY
	Restore registers and return to supervisor (register 8).		R10	

Labels	Chart NU: IPW$$NU - VSE/POWER Nucleus Interval Timer Appendage		Modified Data Fields	Reg. Usage	Calls
	Interval Timer Appendage		Fields	Usage	
	Registers at entry: none				
TI10	Establish base address for timer appendage			R9	
	Calculate base address of nucleus code			R10	
	Set flag that timer-interval expired		CATF(IPW$DPA)		
	Post VSE/POWER master ECB.		PAEB (IPW$DPA)		
	Post VSE/POWER partition dispatchable.			R0,R1, R15	TREADY
TI20	Return to DOS/VSE via exit				EXIT

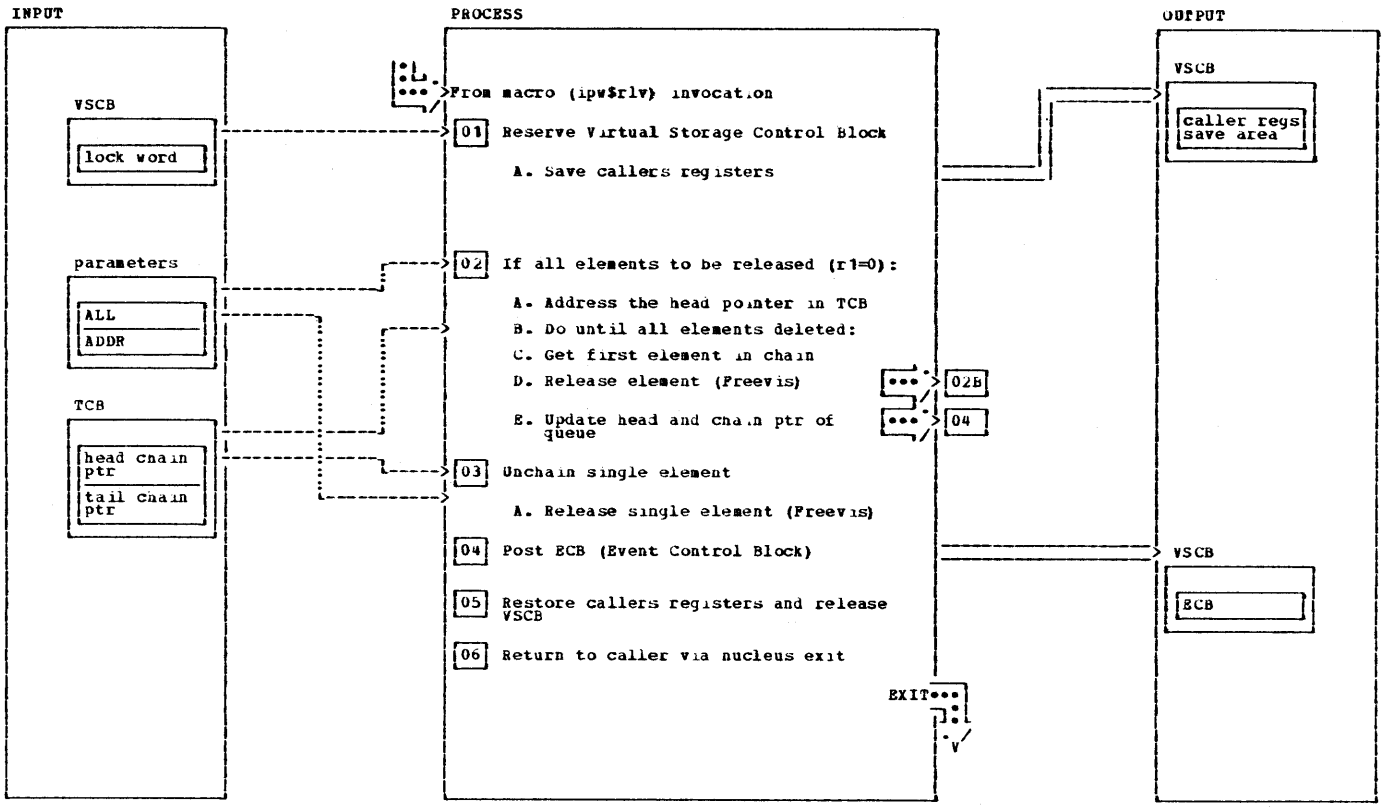
IPW$$NU - Virtual Storage Management



	NOTES	MODULE	LABEL	REF
1	Upon entry register 0 (optional) contains the pool id and a 'no wait' byte, register 1 (required) contains the requested length, and register 4 (optional) start of chain or 'anchor addr'. Test lock word to see if necessary to wait for a Virtual Storage Control Block available. Calling registers are saved in the save area in the Virtual Storage Control Block.			
2	Load register 1 with rounded 128 byte multiple length. (First add buffer header length to user length.) Place pool type in register 2.			
3	Getvis was successful if reg 15 = 0. Test 'TCCB' to see if register 4 contains chain addr (otherwise obtain addr from TCB) and place in reg 1. Chain new element at end of the queue. Update the tail ptr of the queue with the addr of the new entry. Update the forward ptr of the previous entry with addr of new entry. Set forward ptr of new entry to zero and the backward ptr to the addr of previous entry.	GETVIS		

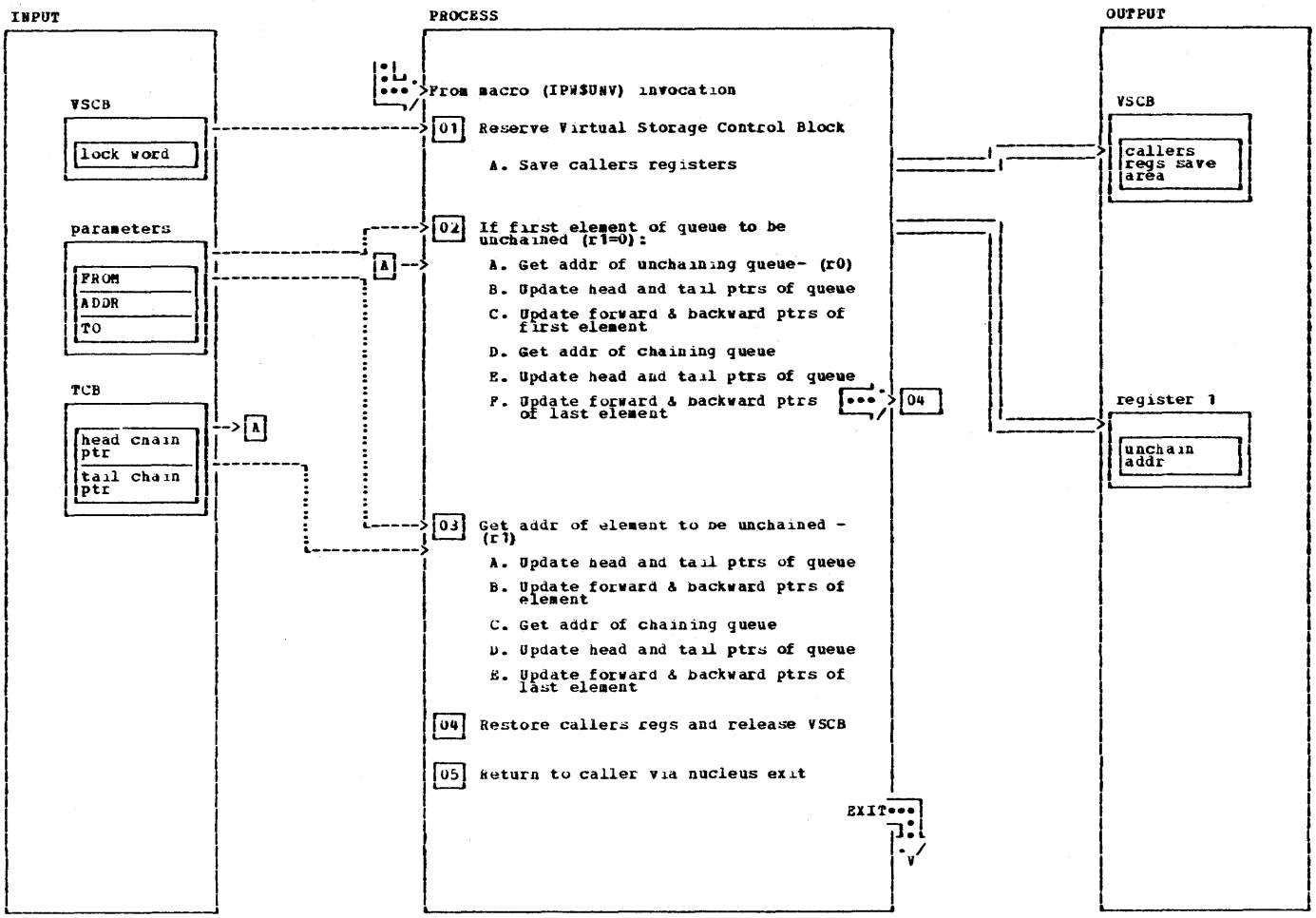
	NOTES	MODULE	LABEL	REF
	Also update the element buffer header with the actual acquired length and the pool id.			
5	If wait has been specified and no space available, then issue (once) message $1Q85I.			$GAM
6	Clear ECB with zeroes before releasing VSCB and restoring registers.			$RLR
	Wait for virtual storage to become available. Try once again to obtain storage.			$WFS

IPW$$NU - Release Virtual Storage



NOTES	MODULE	LABEL	REF
1 Upon entry register 1 (required if 'all' not specified) will contain the address of the queue element to be released and register 4 (optional) will contain the start of chain addr.			
Save reg 2 in reg 0. Issue ipw$rsr macro to release the VSCB. Restore reg 2.			$rsr
Calling registers are saved in the save area in the VSCB.			
2 If reg 1 = 0 then 'all' has been specified. Specify addr to be released in reg 1 and the required length in reg 0. Issue FREEVIS macro	FREEVIS		
Test 'TCCB' to see if reg 4 contains chain addr (otherwise obtain addr from TCB)			
When all entries have been released, update head and tail ptr of queue to indicate empty queue			
3 Test 'TCCB' to see if reg 4 contains chain addr (otherwise obtain addr from TCB)			
Update head & tail ptr of queue & forward and backward ptr of element as follows for unchaining:			
a) If there existed only one element - update head & tail ptrs (BOTH TO ZERO).			
b) If more than one element in queue - update head ptr with the forward ptr of element to be unchained, and the backward ptr of second element (to zero) if we are dealing with the first element.			
c) If we are dealing with a middle element - update previous forward pointer with forward pointer of unchaining element and the backward pointer of following element with the backward addr of unchaining element.			
d) If we are dealing with a last element to be unchained then update tail ptr of queue with backward ptr of unchaining element and update forward ptr of previous element to zero.			
Issue FREEVIS macro.	FREEVIS		
4 Post ECB in the VSCB with X'80' configuration			
5			$rlr

IPW$$NU - Unchain Virtual Storage Element

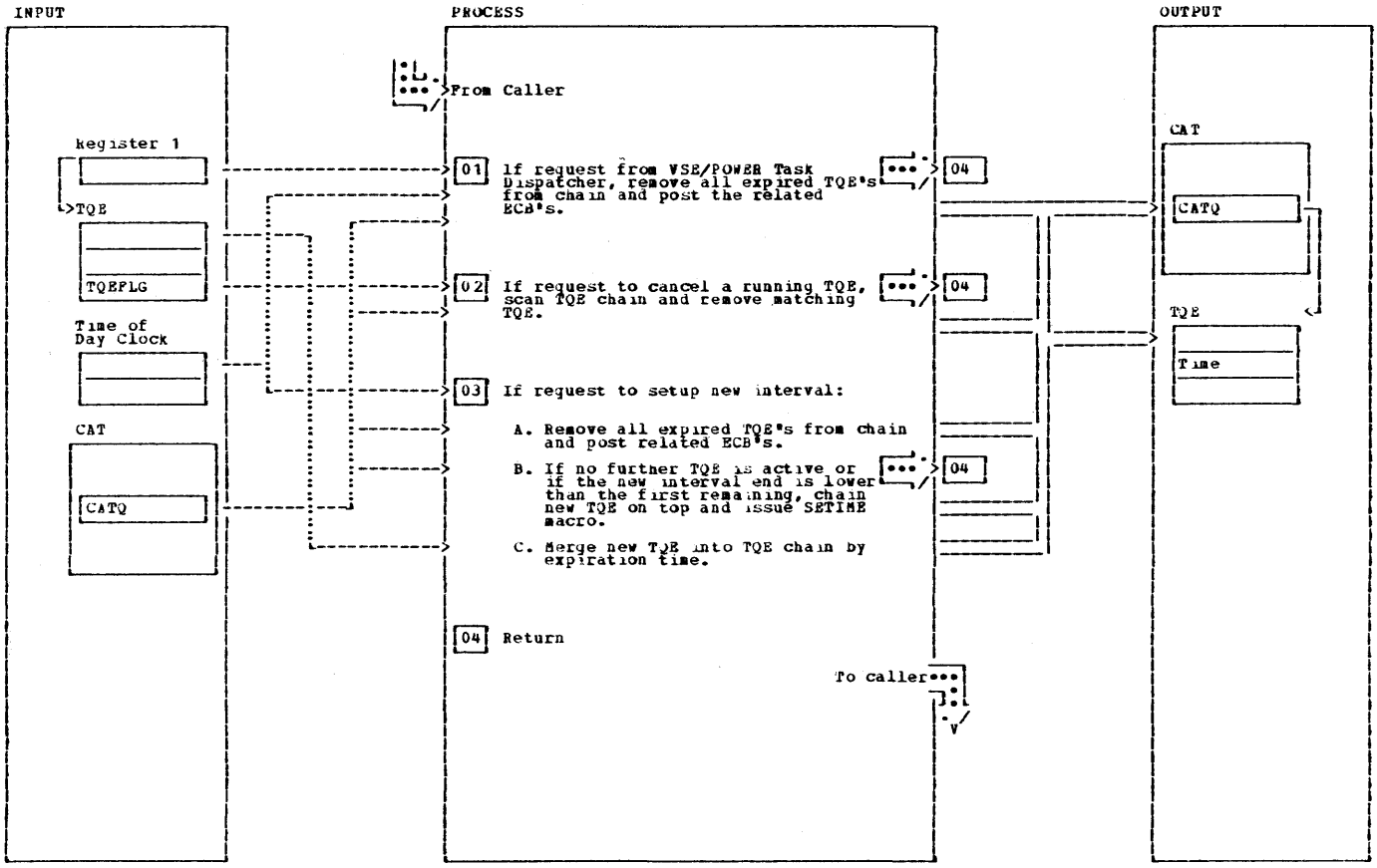


IPW$$NU - Unchain Virtual Storage Element

NOTES	MODULE	LABEL	REF
1 Upon entry reg 0 contains the addr of the head ptr of unchaining queue (required when 'addr' parm not specified), reg 1 contains the adur of unchaining element (optional), and reg 4 contains addr of head ptr of queue for chaining to (optional).			$RSR
2 If 'ADDR' has not been specified (Reg 1 = 0), then the first element of the queue (use 'FROM' parm) is unchained. Return in register 1 the addr of the unchained element. Update head & tail ptr of queue & forward and backward pt of element as follows for unchaining: a) If there existed only one element - update head & tail ptrs (BOTH TO ZERO). b) If more than one element in queue - update head ptr with the forward ptr of element to be unchained, and the backward ptr of second element (to zero). Update head & tail ptr of queue & forward and backward pt of element as follows for chaining: (The chaining element will be placed at end of queue. the head ptr of queue is in reg4 or if not specified - get from tcb). a) If no elements in queue - update head and tail point of queue with addr of new element (head = tail ptr). Update forward and backward ptrs of element (both to zero). b) If more than one element exists in queue update tail pointer of queue with addr of new entry. Also update the former last elements head ptr with this addr. Now update forward ptr of new element (to zero) and the backward ptr of new element with the addr of previous element.			
3 If 'ADDR' has been specified (Reg 1), use this addr for unchaining.			

NOTES	MODULE	LABEL	REF
Update head & tail ptr of queue & forward and backward pt of element as follows for unchaining: a) If there existed only one element - update head & tail ptrs (BOTH TO ZERO). b) If more than one element in queue - update head ptr with the forward ptr of element to be unchained, and the backward ptr of second element (to zero) If we are dealing with the first element. c) If we are dealing with a middle element - update previous forward pointer with forward pointer of unchaining element and the backward pointer of following element with the backward addr of unchaining element. d) If we are dealing with a last element to be unchained then update tail ptr of queue with backward ptr of unchaining element and update forward ptr of previous element to zero. Update head & tail ptr of queue & forward and backward pt of element as follows for chaining: (The chaining element will be placed at end of queue. The head ptr of queue is in reg4 or if not specified - get from tcb). a) If no elements in queue - update head and tail point of queue with addr of new element (head = tail ptr). Update forward and backward ptrs of element (both to zero). b) If more than one element exists in queue update tail pointer of queue with addr of new entry. Also update the former last elements head ptr with this addr. Now update forward ptr of new element (to zero) and the backward ptr of new element with the addr of previous element.			
4			$MLR

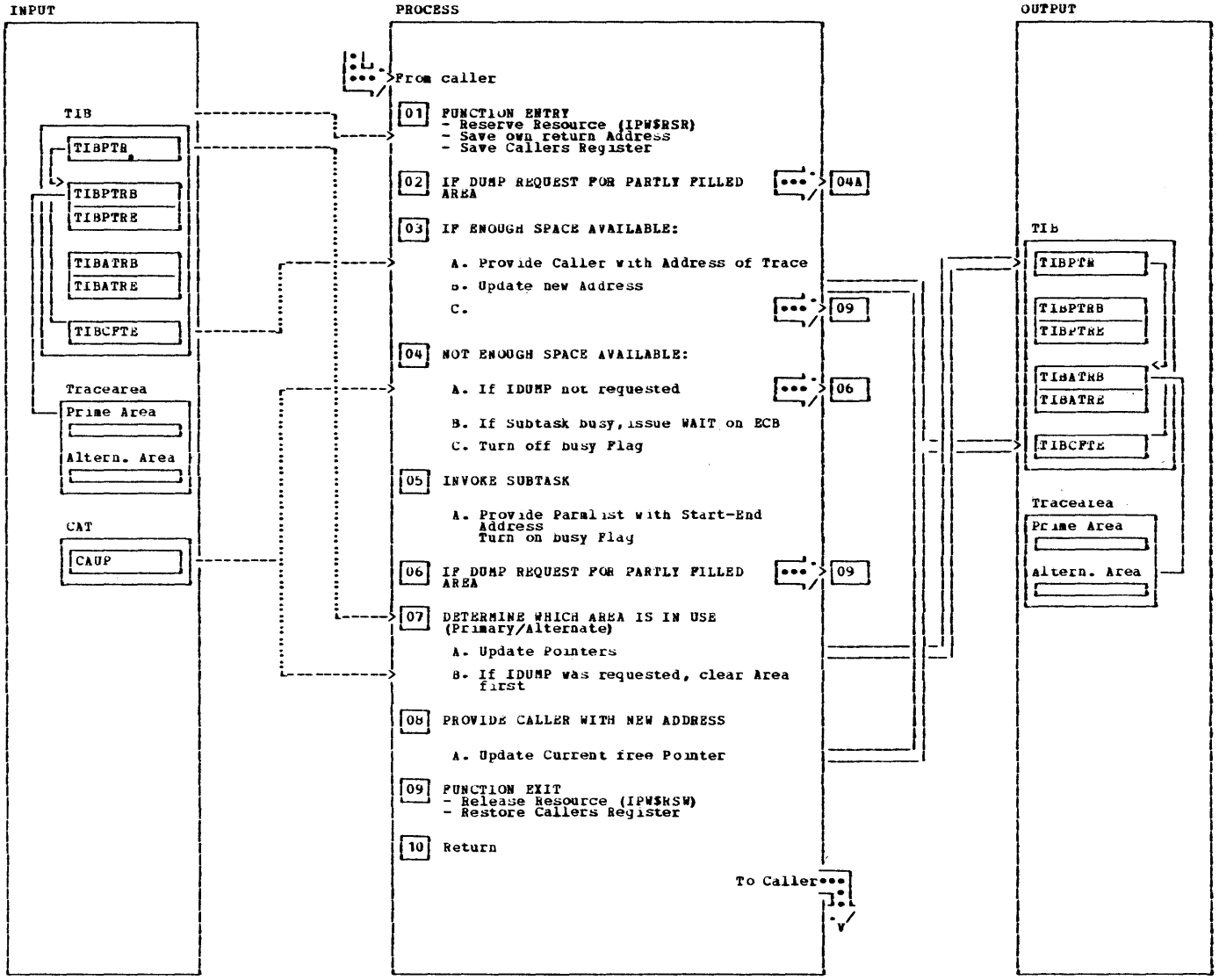
IPW$$NU - VSE/POWER Timer Service



NOTES	MODULE	LABEL	REF
The interval timer service provides an interface between VSE/POWER main tasks and the standard DOS/VSE timer facilities. It will allow multiple tasks to have time intervals concurrently active while maintaining only one timer interval through the SETIME macro. Each interval is represented by a unique timer queue element (TQE). To begin a time interval the VSE/POWER task issues the IPW$STM macro, which (optionally requests TQE-storage and) formats the TQE and then invokes the interval timer service routine. During the interval, the TQE is chained to other active TQE's. Only interval end times are held in TQE's. When the interval expires, the TQE is unchained from the active chain and the related ECB is posted.			
1 Whenever a DOS/VSE timer interval expires, the VSE/POWER Nucleus Timer Exit Routine gets control and indicates the presence of a timer expiration event for VSE/POWER Task Dispatching. The Task Dispatcher activates the Timer Service: The current time of day (TOD) is taken and all expired TQE'S (those TQE's with an interval end time less 1/10 second different from the current TOD value) are deleted from the TQE chain and the related ECB's are posted. A SETIME macro is issued with the new minimum interval length from the first remaining TQE.	SETIME	TS65	

NOTES	MODULE	LABEL	REF
2 Register R1 points to the TQE to be cancelled. The TQE chain is scanned for a matching TQE. If found it is removed from chain.		TS95	
3 At each request to set up a new time interval the current time of day (TOD) is taken and all expired TQE's (those TQE's with an interval end time less 1/10 second different from the current TOD value) are deleted from the TQE chain and the related ECB's are posted. The new time interval is converted into TOD clock units. The interval expiration time is calculated in TOD format and saved in the new TQE. The new TQE is merged into the TQE chain by expiration time: the ordering is lower in front of higher interval end time. If according this ordering the new TQE gets top position, a SETIME macro is issued with the new minimum interval length.	SETIME	TS25	

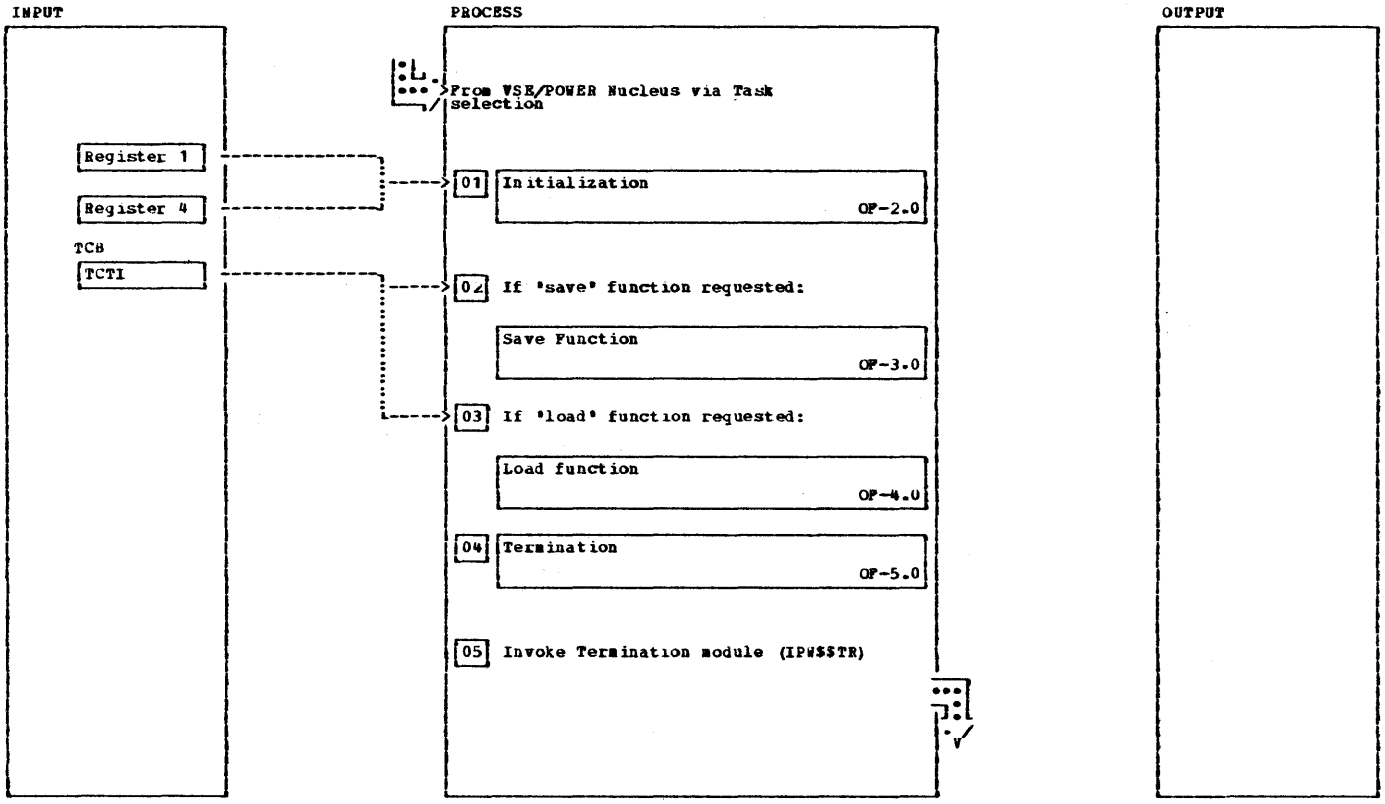
IPW$$NU - GET TRACE ENTRY ROUTINE



NOTES	MODULE	LABEL	REF
2 The length of zero indicates that the partly filled Area is to be dumped if IDUMP was requested.			
3 The Caller provides in R0 the required Recordlength. A check is made if there is enough space in the current Area (Primary/Alternate).			
3A The Caller will get the Address where he can write his Record in R1.			$GTE
3B The next free Address is stored in the Current Free Pointer in the TIB.			
5 The Subtask is called via: IPW$IAS TYPE=SERVICE,TASK=DUMP			
5A The Parmlist in the SRB is provided with the Start and End Address of the current Area which is to be dumped.			

NOTES	MODULE	LABEL	REF
6 If just the partly filled Area was dumped, no updates are necessary.			
7 A check is made which Area is in use now, then the Swap to the 2nd Area is performed.			
7A The TIBPTR as well as the TIBCFTE are set to point to the the 2nd Area.			
7B The Area is cleared first if IDUMP was requested. Otherwise a wrap around is performed.			
8 The new Address is stored in the Register save Area so that on return the Caller has this Address in R1.			
9 The Asynchronous Service Block is released and the callers Register restored.			$NLR

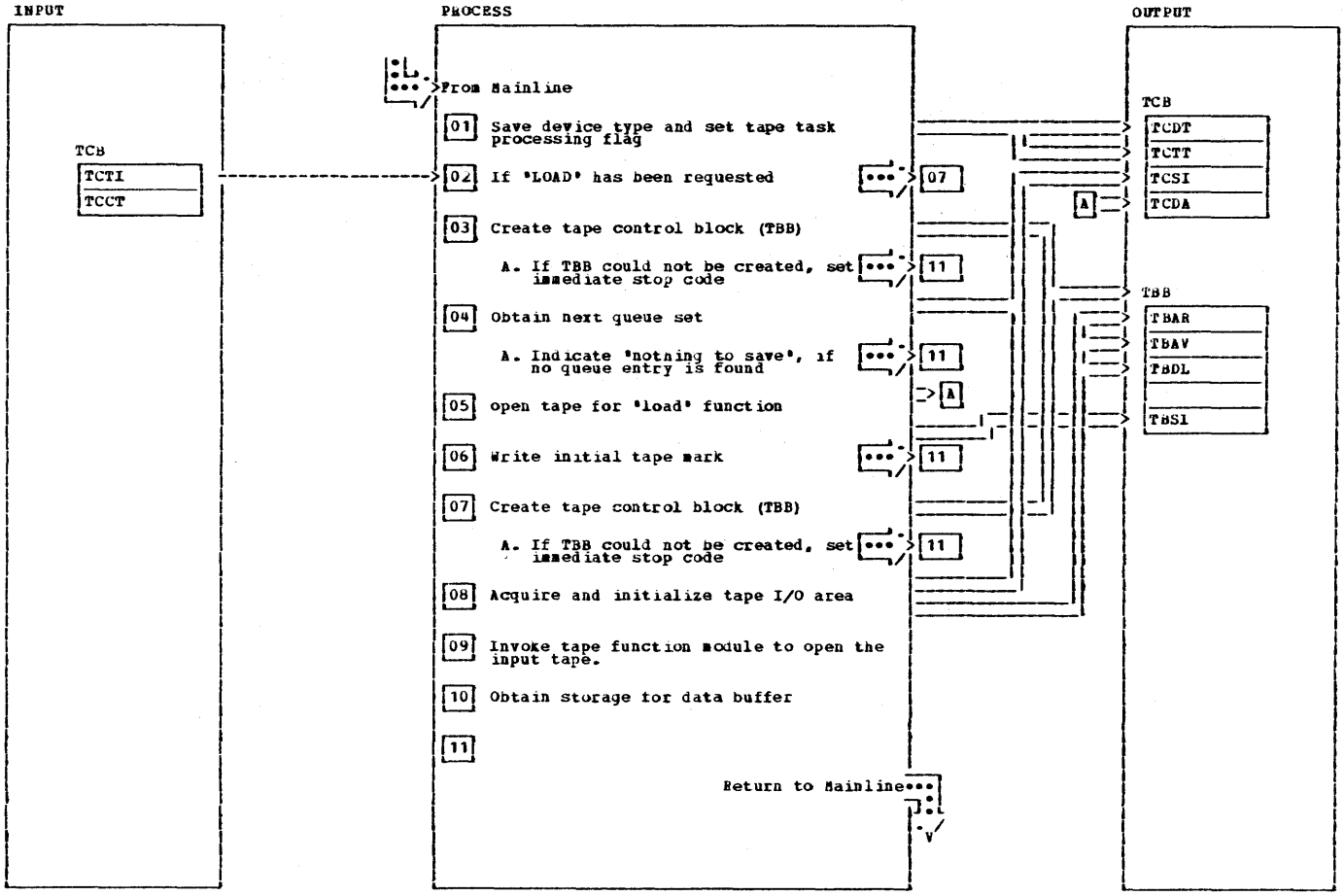
IPW$$OF - M A I N L I N E



NOTES	MODULE	LABEL	REF
1 If any error detected during initiation, the termination code in the TCB, field TCTT is set to stop immediately.			
2 The function code already set from the command processor module (IPW$$CM) during command and parameter checking is tested to find either 'LOAD' or 'SAVE'.			

NOTES	MODULE	LABEL	REF
5 Instead of immediate detach of the task, the termination module (IPW$$TR) is invoked to release all storage and may be control blocks and finally detach the task.			

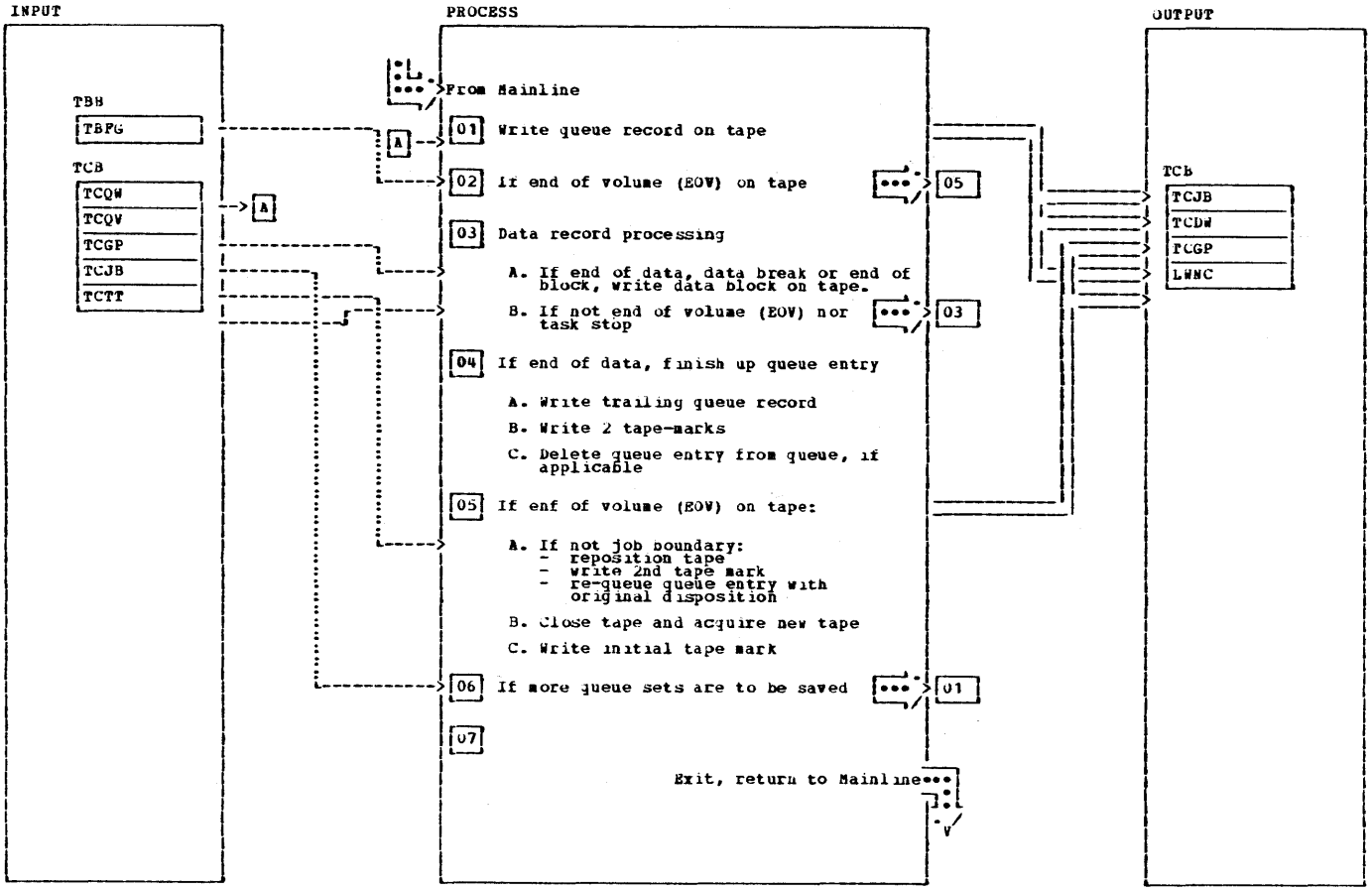
IPW$$OF - INITIALIZATION



	NOTES	MODULE	LABEL	REF
1			OFI0	
3	The module IPW$$OT is invoked via the IPW$OTP function macro call to create a tape control block (TBB) for output processing.			$OTP
4	If a TBB was created succesfully, the Queue management is invoked to obtain a queue set according to the class specified within the POFFLOAD command. If no queue set is eligible in the class chain currently searched, the next class entry is addressed and the 'get next qeue set' function is re-called. This process continues until all class(es) are searched. If not any queue set available, saving is not initiated.			$GQS
5	The module IPW$$OT is invoked via IPW$OTP function macro call to open the tape. (output processing).			$OTP

	NOTES	MODULE	LABEL	REF
6	One tape mark is then written.			$CCT
7	The module IPW$$OT is invoked via IPW$OTP function macro call to create a TBB for input processing.		OFI3	$OTP
8	A work area with a length of 2008 is initiated to allow reading of the maximum data size. The virtual and if necessary the real address are stored in the TBB.			$RSW
9	The module IPW$$OT is invoked via IPW$OTP function macro call to open the tape for load purposes (input processing).			$OTP
10	A data buffer is reserved to hold the data records being passed via loading takes place.			$RSW

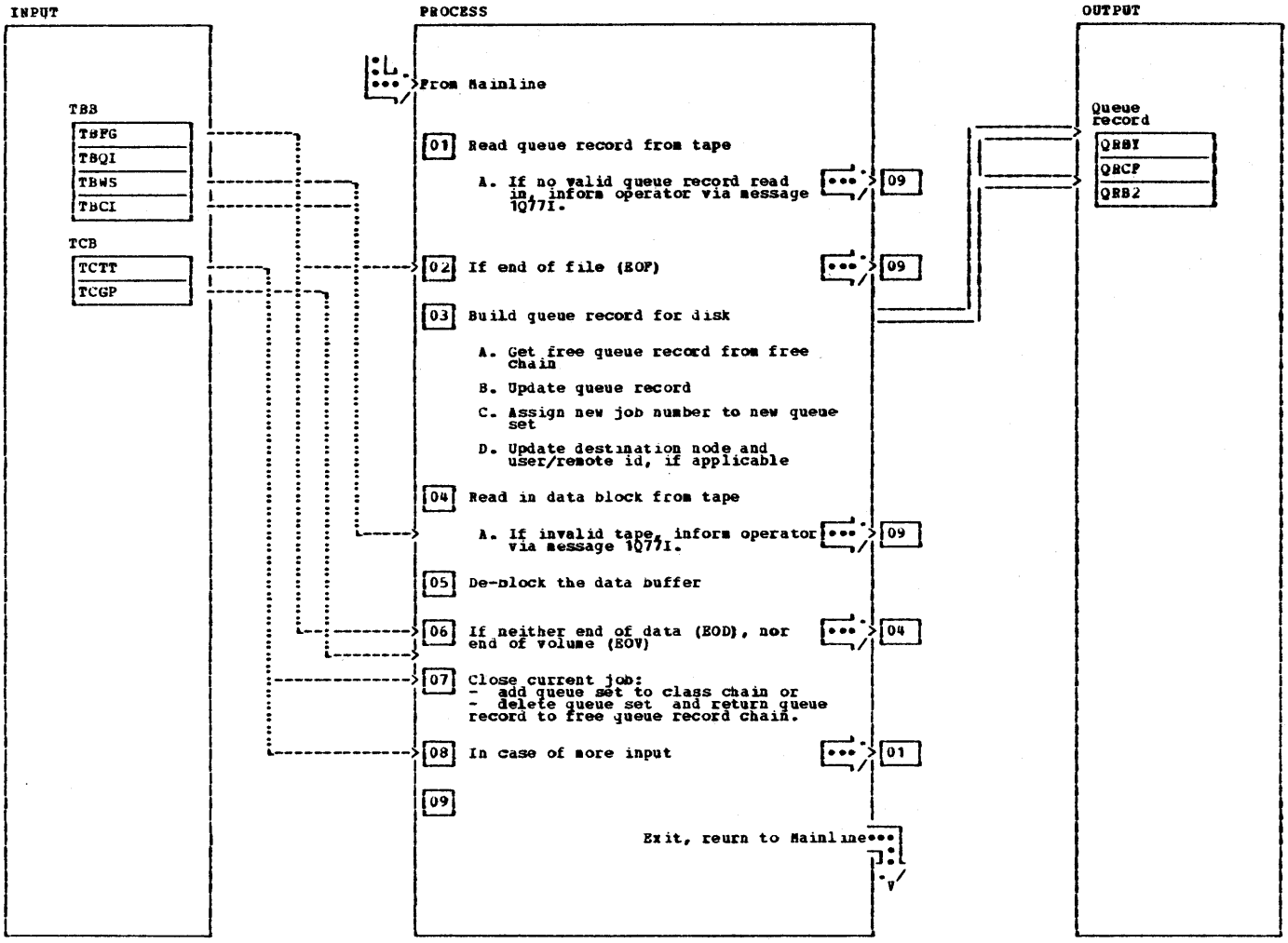
IPW$$OF - S A V E FUNCTION



	NOTES	MODULE	LABEL	REF
1	The queue record obtained via queue management function GQS is written on tape, this is in fact first queue record within a queue set.		OFSO	$WTT
3	A data record is made available via the data management function GET DATA RECORD from a data block.		SDBO	$GDR
	In case of end of logical data block, the total block is written on tape similiar to the disk function.			$WTT
4	The trailing queue record (queue record currently held by the task) is written on tape.		SQWO	$WTT
	After the queue record has been written sucessfully two tape marks are written to close the queue set currently being saved.			$CTT
	The tape is repositioned behind the first tape mark to allow a new queue set to be saved.			$CTT
	The queue set being processed is dequeued according to its disposition.			$DQS

	NOTES	MODULE	LABEL	REF
5	If the task is on job boundary, the close tape function within IPW$$OF module is invoked.			$FQS $OFP
	If the task is not on job boundary, the tape is repositioned to the last queue set being processed sucessfully.			$CTT
	In case a new tape is required to continue, the task is forced to do so by resetting the termination type in the TCTT field of the TCB to X'40'.			
6	If no stop code is set in the TCB nor end of volume indicated in the tape control block the 'get next queue set' function is called to obtain the next eligible queue set to be offloaded.			$NQS
	If no queue set is eligible, the next class entry is addressed, if applicable and the 'get next queue set' function is invoked again.			

IPW$$OF - L O A D FUNCTION

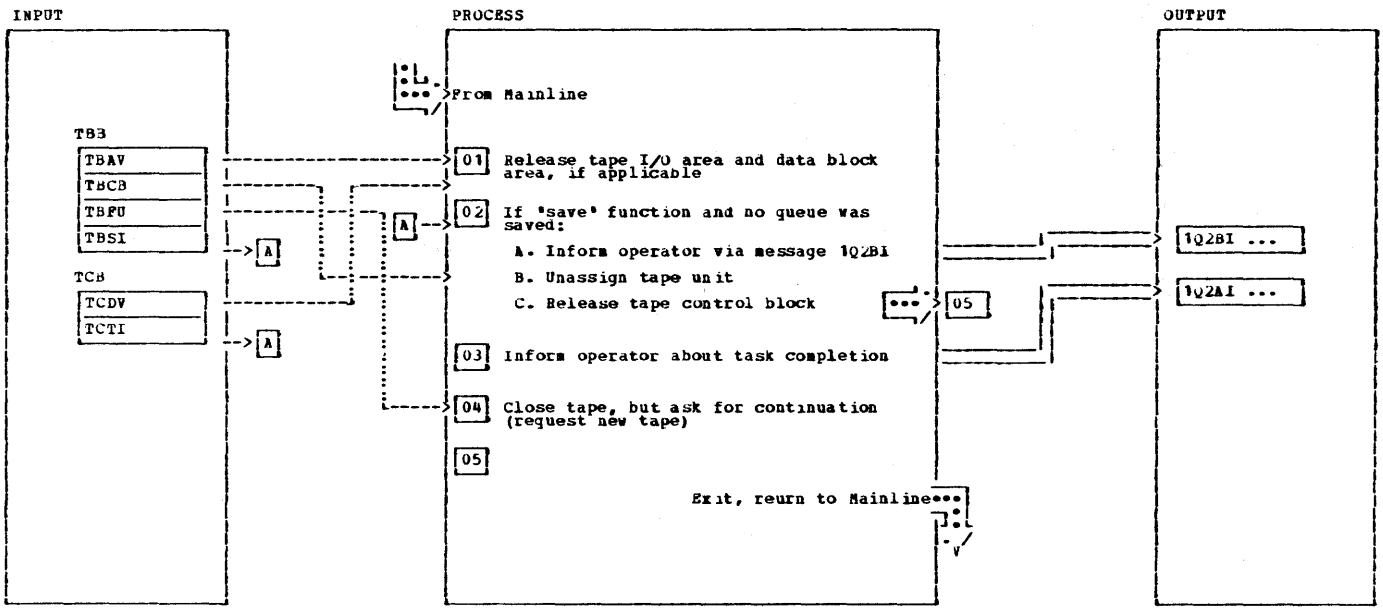


IPW$$OF - L O A D FUNCTION

	NOTES	MODULE	LABEL	REF
1	The tape is read and the record being read is checked to find a valid queue record according to the POFFLOAD parameter (queue-id).		RQR0	$RDT
	In case an invalid record has been found the operator is informed via message:			$GAM
	1Q77I INVALID SPOOL TAPE task,cuu			
	and the task is forced to stop.			
3	A free queue record is requested from the queue management function 'Reserve queue record' and the queue record is updated with information obtained from the queue record read in from tape.		RQR0	$RQS
	The disk management block (DMB) is locked during the update of the new job number.			$RSR
3				$RLR
4	During a data block cycle, a data block is read from tape and the logical data blocks are passed in turn to the data management		RDR0	$RDT
	Since the tape may have been written with a different DBLK size the first block is read with the maximum data block size supported by VSE/POWER. The reasidual count is then used to calculate the actual data block size. In case an invalid record has been found the operator is informed via message:			$GAM
	1Q77I INVALID SPOOL TAPE task,cuu			
	and the task is forced to stop.			

	NOTES	MODULE	LABEL	REF
5	Each data record is now passed to the 'put data record' routine. If the record is an extended record, the record is re-constructed in its original format before passing it to the 'put data record' routine.			$PDR
	This repeats as long as data blocks are read from tape.			$RSV
7	The queue set including queue records and the data records belonging to it are added to the queue file by using the function: 'ADD TO QUEUE'.			$AQS
	If however, meanwhile any termination code is set in the TCB (except of 'E'), the queue records occupied of the queue set are returned to the free queue record chain.			$DQS
	The tape is skipped behind the trailing tape mark, ready to read another new queue record or to find end-of-file condition.			$CTT
8	This process continues when either no termination code or the flush termination code was set in the TCB			

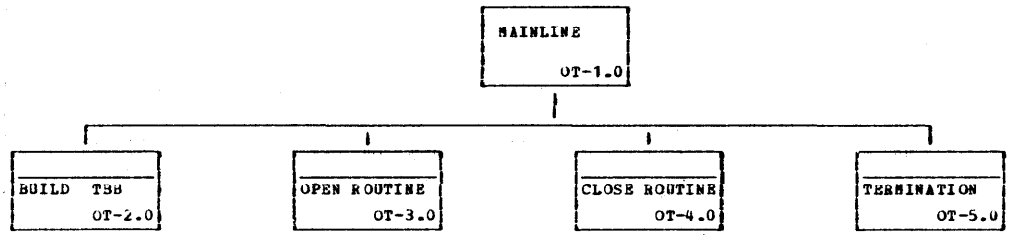
IPW$$OF - TERMINATION



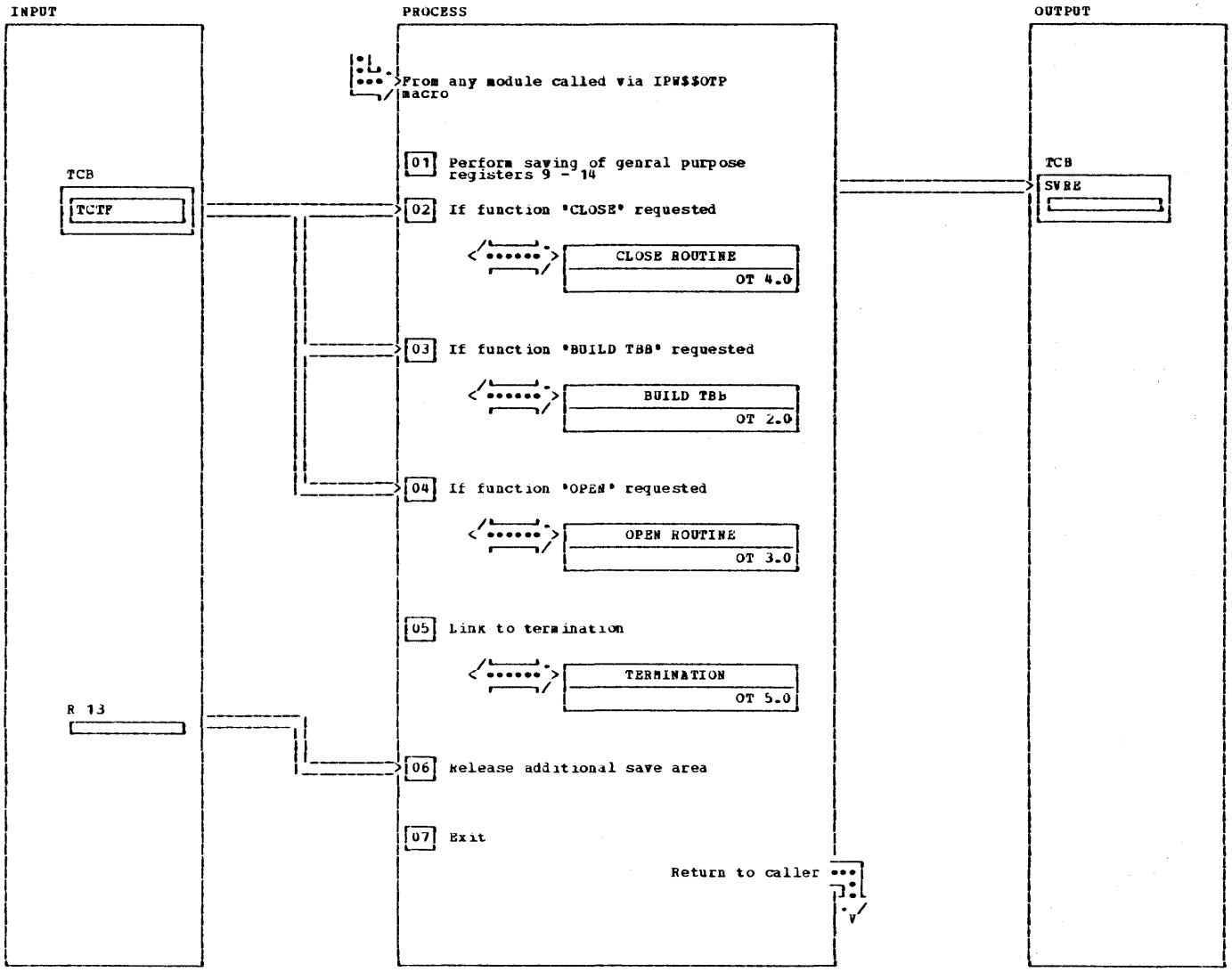
NOTES	MODULE	LABEL	REF
2B The tape unit being assigned at task start time is released from the VSE/POWER partition.			$ULP
3 The POFFLOAD task termination message is written to the console and informs the operator that the task has been completed. 1Q2AI OFFLOADING SUCCESSFULLY COMPLETED			$GAM

NOTES	MODULE	LABEL	REF
In addition to this, the module IPW$$OT is invoked to close the tape function and to release the tape from the VSE/POWER partition when the task normally ends.			$OTP

CHART OT: IPW$$OT



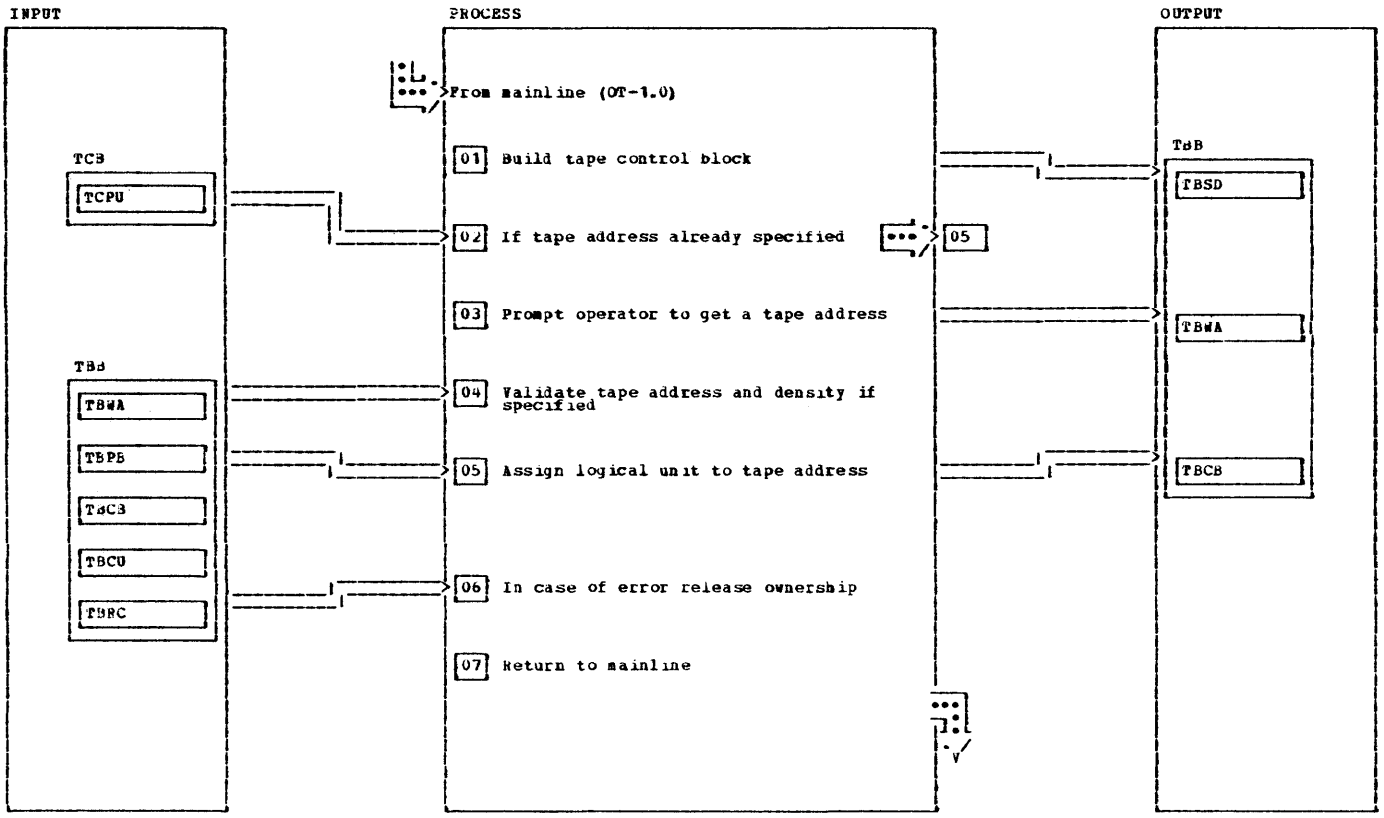
IPW$$OT - M A I N L I N E



NOTES	MODULE	LABEL	REF
1 The caller registers 9 - 14 are saved into the TCB function save area.		OT00	$SAV
An additional save area is created allow saving of the registers 9 - 14 if another function is invoked.			$RSW

NOTES	MODULE	LABEL	REF
6 The additional created save area is released		OT14	$RLW
7 The callers register 9 - 14 are reloaded from the TCB function save area and control is passed to the caller			$RET

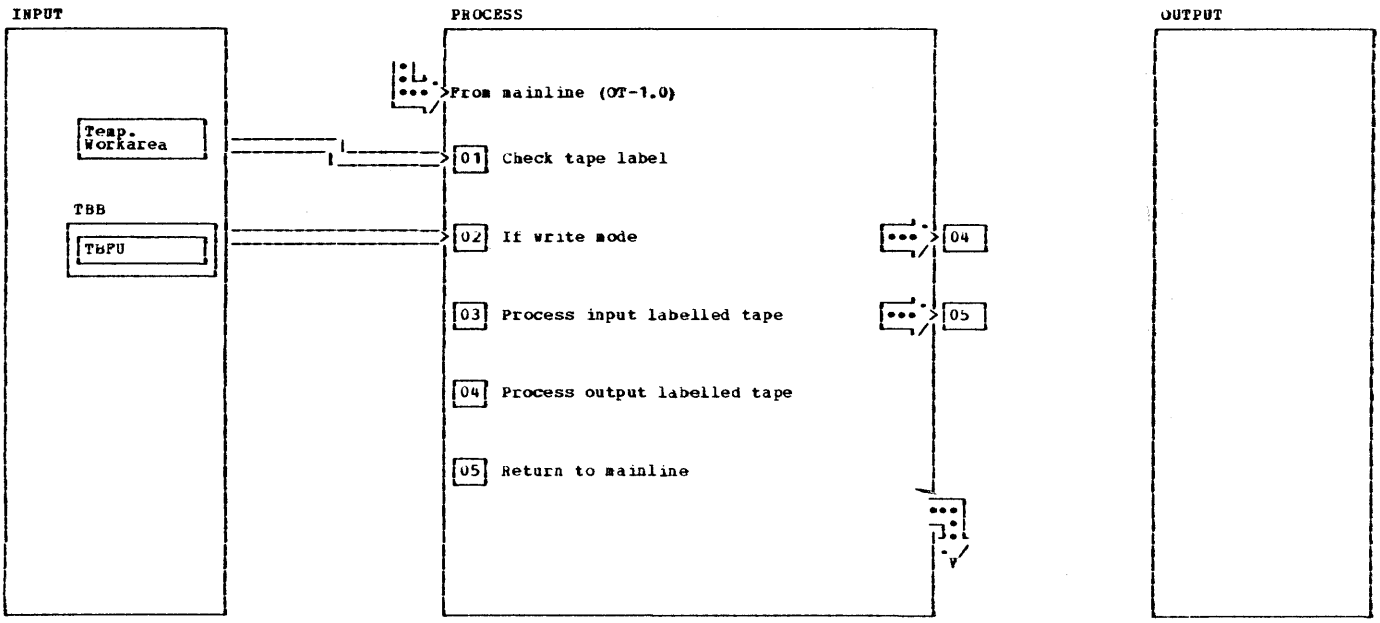
IPwSSOT - BUILD TBB



NOTES	MODULE	LABEL	REF
1 Real storage is reserved to create a tape control block (TBB) related to the task which has initiated the function. The TBB is described and CCB and common CCW is created.			$RSW
3 The operator is prompted to specify a tape address and may be tape density if necessary.		OT20	$WTR
4 The tape address issued by the operator is syntax checked. The tape density if specified is validated and if valid saved in the TBB.			

NOTES	MODULE	LABEL	REF
5 The tape address is used to find an unowned Physical Unit Block (PUB) and assign it to a free Logical Unit Block (LUB).			$ULP
6 If no free LUB is available the PUB is released from the ownership of VSE/POWER			$ULP

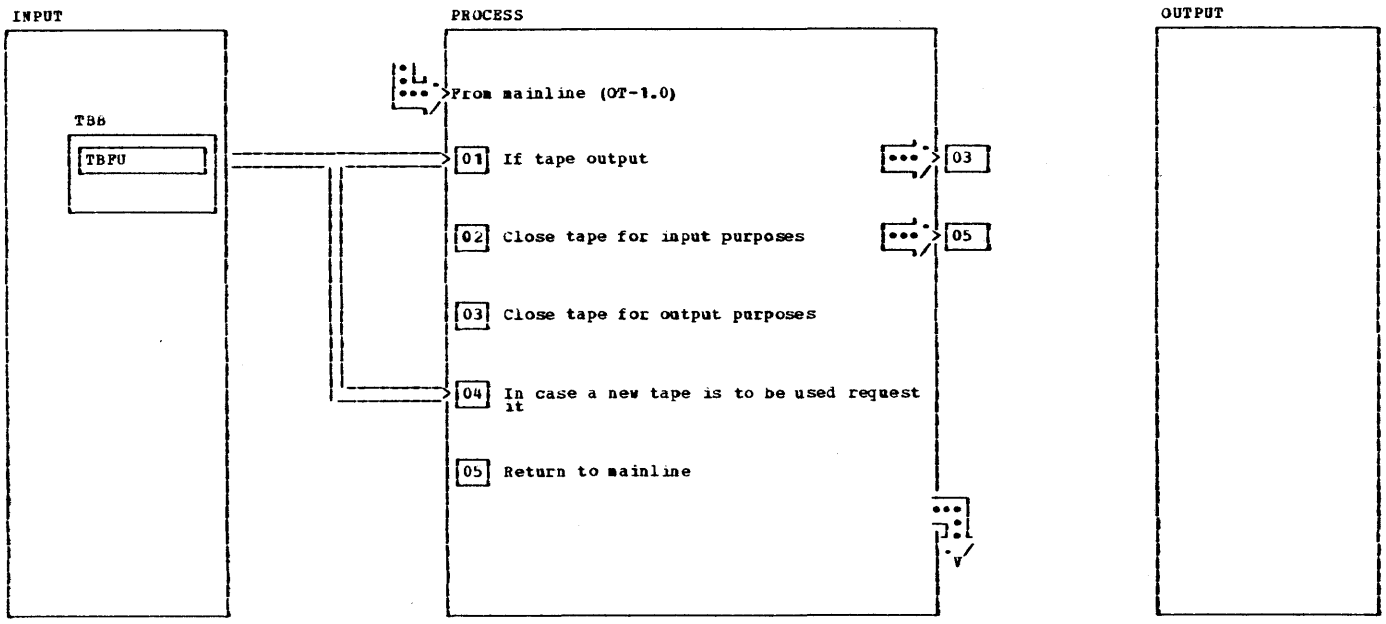
IPW$$OT - OPEN ROUTINE



	NOTES	MODULE	LABEL	REF
1	In case a workarea is available it is taken to read the tape and check for standard labels otherwise a temporary workspace is used.			$HSW
	The tape unit is sensed to get loadpoint, protection ring, and mount information.			$CTT
	The first record being read is checked whether this is a 'VOL1 header record or not. If not, the tape is marked as unlabelled tape.			$RDT
3	If a 'VOL1' label was found another record is read to find a 'HDR1' record. If either a valid header label found or an unlabelled tape to be processed the first data record is read.			$RDT
	In case a valid header label was found, the operator is informed to verify for a correct tape being mounted.			$WTO

	NOTES	MODULE	LABEL	REF
	The data record being read is used to determine the length of it and to analyze and verify the tape input in relation to the task calling this function. The TBB is updated to inform about record length, blocking factor, labelled tape.			$WFO
4	The tape being mounted is label checked to find whether this is an unlabelled tape or not.			$RDT
	If not the tape is repositioned to loadpoint, available to be used for output purposes.			$CTT
	If a labelled tape being mounted the operator is informed to have a change to save a maybe protected tape.			$WTO
	If a density was specified the default density in the PUB is exchanged with the specified one.			$WFO

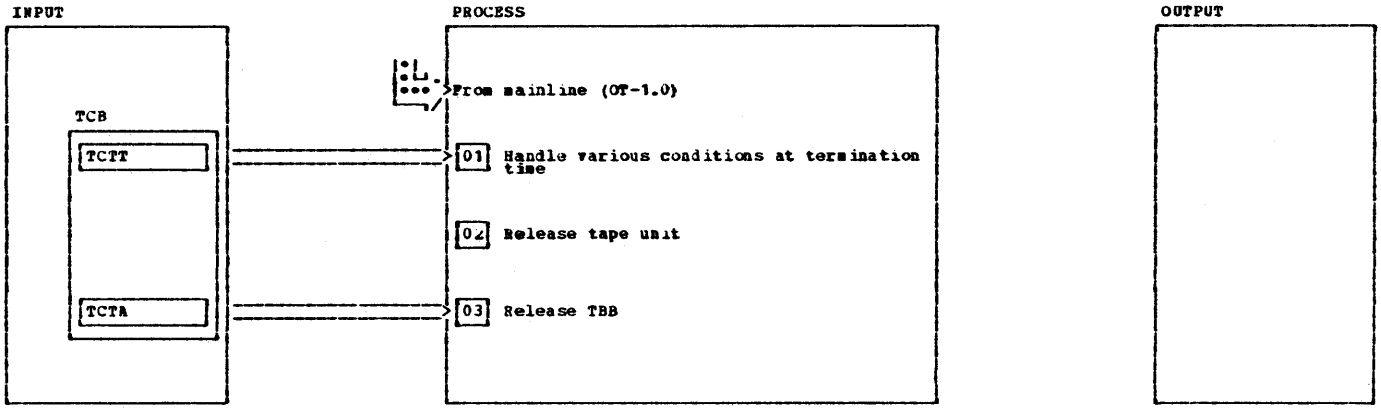
IPW$$OT - CLOSE ROUTINE



NOTES	MODULE	LABEL	REF
2 The tape is read to find either a trailer label in case a labelled tape is being processed or to find a file marker if unlabelled tape being processed.			$RDT
If an 'EOF1' trailer record is found the tape is forced to end-of-file and ready to get another tape file of a multifile volume. If a multifile volume the new file is opened be using the open tape routine.			$CTT
If an 'EOV1' trailer record is found the tape is unloaded and the subsequent tape is requested for processing.			$WTO

NOTES	MODULE	LABEL	REF
3 If the tape is used to spool output data from a partition the tape is neither unloaded nor rewound, it is keeped for subsequent jobs to be processed. If not a spool tape affected, the tape is unloaded.			$CTT
4 The operator is informed via message:			$WTO
1R58I MOUNT TAPE ON cuu FOR jobname jobnr task-id			
to mount a new scratch tape and to reply for the tape after being mounted.			

IPW$$OT - TERMINATION

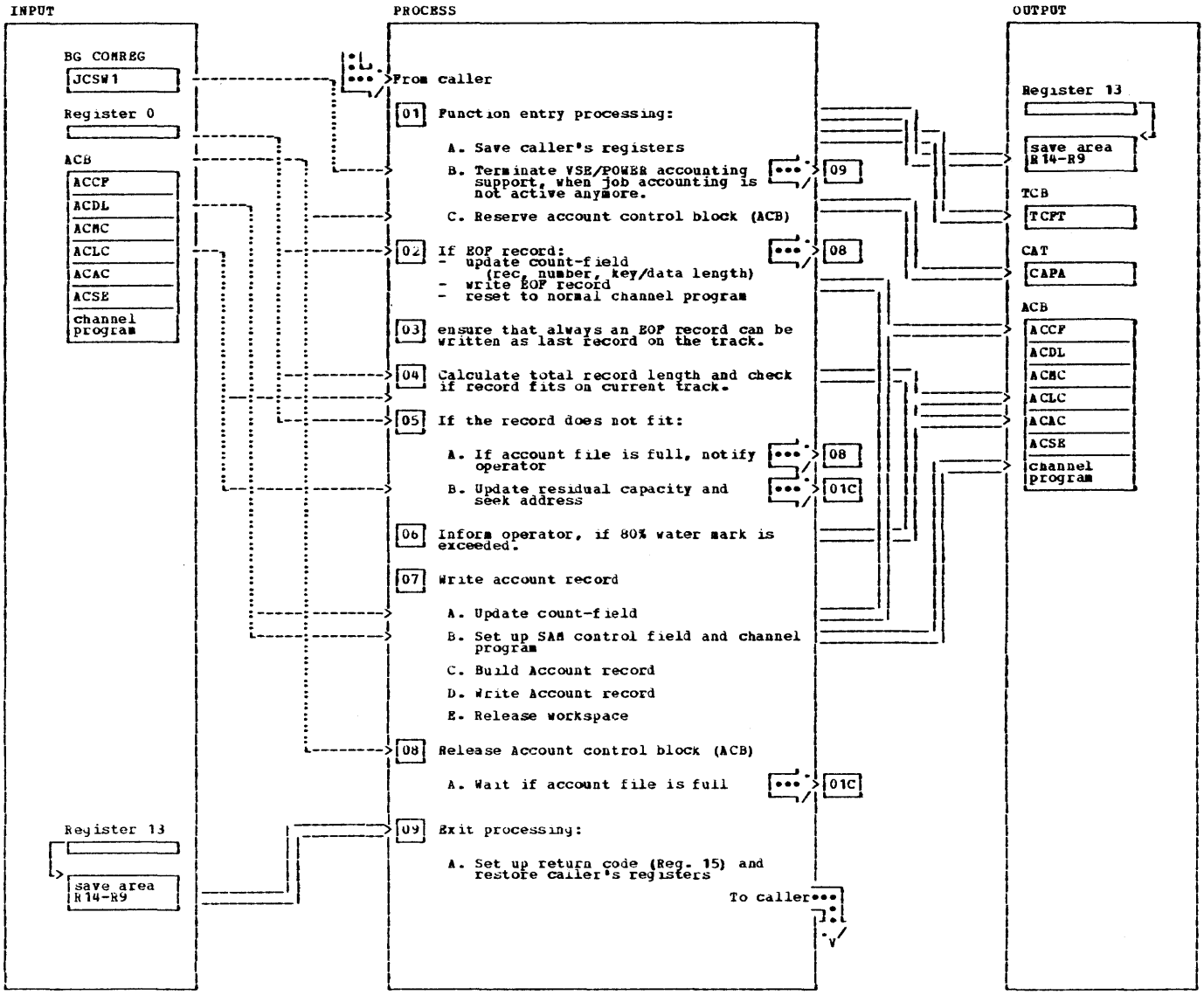


NOTES	MODULE	LABEL	REF
1 If a temporary workspace was owned during open it is released now. If an execution writer to be stopped/flushed the disposition is changed from 'T' to 'D'.			$RLW

NOTES	MODULE	LABEL	REF
2 The input tape is being closed, the the tape is unloaded and the LUB is unassigned from the related PUB.			$ULP
3 The storage owned by the TBB is released and the address pointer in the TCB is set to zero.			$RLW

This page was left blank intentionally.

IPW$$PA - Put Account record



IPW$$PA - Put Account record

NOTES	MODULE	LABEL	REF
1 The caller's registers are saved in the save area addressed by register 13. The function track byte in the TCB of the calling task is set to 'L' to indicate that put account is active. The task identifier in the TCB is examined if the account record should actually be written. If it is still initialization time (autostart processing), or the task is a 'save account' task, or the task is a 'print status' task creating a LST queue entry, control is passed back to the caller.			$SAV
1B The BG communication region is examined if job accounting is still active. If not, the operator is informed via message 1084I ACCOUNT SUPPORT CANCELLED and the entry point address of the put account routine in the CAT is cleared to indicate no accounting support.			$GAM
1C The Account control block (ACB) is reserved for the duration of the function.			$RSR
2 The account record length is examined if an EOF record (indicated by a zero record length) should be written. If so, the current record number in the count field is updated (incremented by 1) and the key/data length is set to zero. The channel program is modified to write just the count field. On completion of the I/O the command and data chain flags are set again and the sector value is set for the next cycle.	EXCP	PAEF	$WFC
3 The maximum track capacity is updated (decremented) by the amount of bytes necessary to write an EOF record. Is is done to ensure that always an EOF record can be written as last record on the tracks.		PA110	
4 The total record length contains the original record length, the 8-byte block field length plus the optional standard prefix size, if applicable.		PA120	
A check is made to examine if the new record fits on the current track.	GETVCE		
5 Not enough space is left on the current track to write the account record.		PA200	
5A Message 1032I NO MORE ACCOUNT FILE SPACE .... is issued to the operator to inform him that the account file is full, when the upper limit has been exceeded (current track is last track of the account file). Register 4 is set to zero to indicate the account file full condition.			
5B The residual total capacity is decremented by the (now useless) residual track capacity of the current track. The seek address is updated to address the first record on the next track/cylinder.			
6 If the residual capacity is less than 20% percent of the total account file capacity and the operator has not yet been informed, message 1031I MORE THAN 80% FULL ACCOUNT FILE ... is issued. The high order bit of the 20 percent limit is set to 1 to bypass the above comparison on the following occasion.			$GAM
7A The current record seek address (CCHHR) is moved to the count field and the record number is incremented by one to point to the next account record.		PA160	
7B Workspace is reserved to build the account record, whose virtual address is stored in the ACB and whose real address is stored in the write data CCW.			$WSW
7B The length of the total account record is stored in the block field.			
7C The Account record is constructed from the SAM block field, the optional VSE/POWER standard prefix and the account record as passed by the caller. The record is built in the workarea prior acquired.			
7D	EXCP		$WFC
7E			$WLW
8 The account controlblock is released.		PAEX	$RLR
8A If the account file is full (Reg. 4 contains zero) the account event control block is reset, ans the task is put in internal wait state. On return from this wait state after the Account file has been saved, a branch is made to restart writing the account record.			$WFC
9 The caller's registers are restored from the save area addressed by register 13 and return is made to the caller The function track byte in the TCB is reset to indicate that the put account function is finished. Additionally register 15 is set to zero (special requirement for the task terminator routine (IPW$$TH).			$RET

IPW$$PD - VSE/POWER Put Data Record	
Label	Routine
PD00	Function Entry
PD01	Data Record Blank Suppression
PD08	Check for Space Available in Buffer
PX02	Extended Length Record Processing
PD12	Create Data File Block Record
PD40	Increment Disk Address Subroutine
PD70	Write Block to Disk Subroutine
PDRT	Function Exit

Services Used	
Service	Macro
Task Management	IPW$WFC
Resource Management	IPW$RLR IPW$RSR
Message Service	IPW$GAM
Disk / Tape Service	IPW$RDQ IPW$WTD IPW$WTQ IPW$WTT

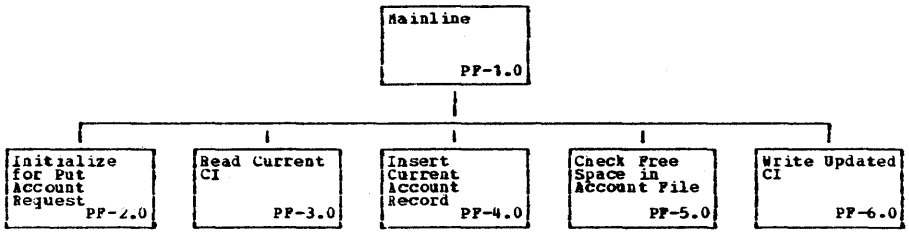
Called by IPW$PDR	
Module	Description
IPW$$LR	Logical Reader
IPW$$NR	Network Receiver
IPW$$OF	Offload Queues
IPW$$TR	Task Terminator
IPW$$XW	Execution Writer

Labels	Chart PD: IPW$$PD - Put Data Record		Modified Data Fields	Reg. Usage	Calls
PDSD	The first 16 bytes constitute the section descriptor: 'PDCS release'				
PD00	Function Entry:				
	Save registers 14 through 9 inclusive		IPW$DSV		
	Set addressability for DMB.				
	If end-of-block posted, branch ....>	PD19	TCGP(IPW$DTC)		
	If caller's data length > 32K-1 bytes then set data length to 32K-1.			R9	
	If caller's data length = 0, ...>	PD78			
	Data Record Blank Truncation:				
	If data record is an internal control record (TCCC=x'ff'), skip blank truncation .....>	PD07	TCCC(IPW$DTC)		
	Skip truncation of trailing blanks in record if the data is for the MFCU card reader and is: - a write command (x'45') - a load print buffer command for buffers 1-6.		TCRL(IPW$DTC) QRDT(IPW$DQR)		
PD05	Truncate trailing blanks in blocks of four bytes.				
PD07	If the caller is the execution writer, and has been cancelled then return .....>	GDRT	TCTI(IPW$DTC) SVR7(IPW$DSV) SVR4(IPW$DSV)		
PD08	If the data record will fit into the data block record, branch .....>	PD12	DRDL(IPW$DDR) BLDB(IPW$DPA)		
	If the data record is not an extended record (>=DLBL-8 bytes) then go write buffer .....>	PD30			
	Extended Length Record Processing:				
	If room exists in the present data block for data, branch .....>	PX02			
	Otherwise, set end-of-block in the previous data record, and then link to write the data block and update disk I/O request word.	PD70 PD40			
PX02	Indicate first extension record.		DRG3(IPW$DDR)		

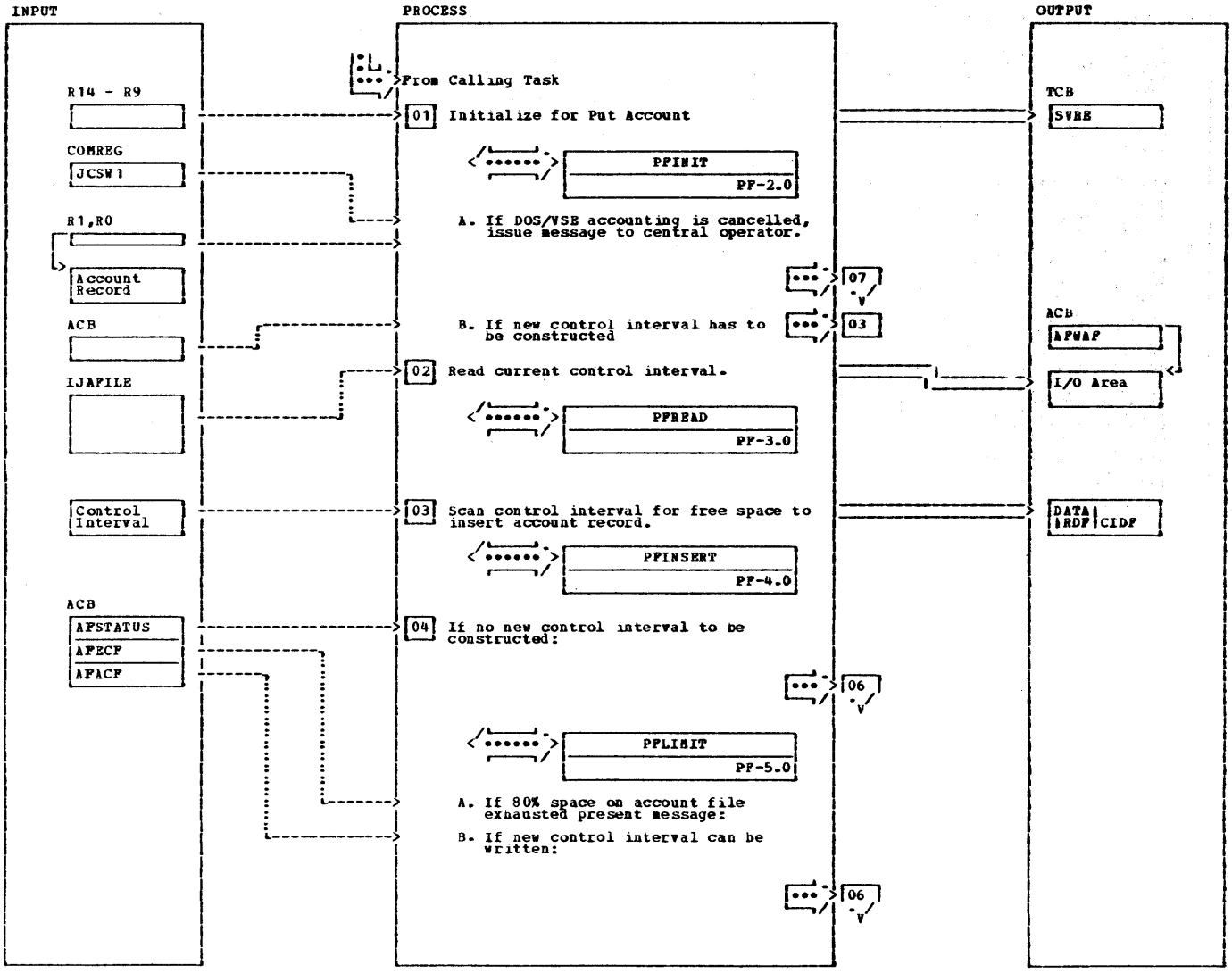
Labels	Chart PD: IPW$$PD - Put Data Record		Modified Data Fields	Reg. Usage	Calls
PX04	Set up data block control word:		DRDS(IPW$DDR)		
	- data block record length		DRRL(IPW$DDR)		
	- general purpose byte (1)		DRGP(IPW$DDR)		
	- command code		DRCC(IPW$DDR)		
	- general purpose byte 2		DRG2(IPW$DDR)		
	- general purpose byte 3		DRG3(IPW$DDR)		
	- extended record residual byte count		DRG3(IPW$DDR)		
	Indicate end-of-block and extended record.		DRGP(IPW$DDR)		
	Move the data extension to the data block record.			R6,R7 R8,R9 R0	
	Link to write the data block and update the disk I/O request word.	PD70 PD40			
	If this is the last extension ....>	PX06			
	Indicate this is a middle extension and loop back to process ....>	PX04	DRG3(IPW$DDR)		
PX06	Move general purpose byte 1 to data.		TCGP(IPW$DTC)		
	Indicate last data record extension.		DRG3(IPW$DDR)		
	Indicate extended data record.		DRGP(IPW$DDR)		
	Branch to ....>	PX08			
	Create Data BLock Record:				
PD12	Set up data block record control word			R3	
	- general purpose byte (1)		DRGP(IPW$DDR)	R3	
	and reset next-record pointer.		TCPR(IPW$DTC)		
PX08	Continue set up of data block record control word:				
	- data block record length		DRRL(IPW$DDR)		
	- command code		DRCC(IPW$DDR)		
	- general purpose byte 2		DRG2(IPW$DDR)		
	Move (remaining) data to data block record.			R6,R7 R8,R9	
	Store remaining capacity in current block.		TCBC(IPW$DTC)		
	If it is not a VSE/POWER control record and not a reader task and it is end-of-data, do not count the record....>	PD25			
	Update record counter.		QRNR(IPW$DQR)		
PD16	If data break record....>	PD20			
	If end of data record....>	PD25			
PDRT	If normal record, restore registers and return to caller.			R14-R9	

Labels	Chart PD: IPW$$PD - Put Data Record		Modified Data Fields	Reg. Usage	Calls
	Handling of Special Conditions				
PD19	Unexpected end of input:				
	Empty buffer.....>	PDRT			
	Indicate end of block in previous record if available		DRGP(IPW$DDR)		
PD20	Data break:				
	Write data block.....>	PD70		R14	
	Update disk request word.....>	PD40		R14	
	Return to caller.....>	PDRT			
PD25	End of data.				
	Write data block.....>	PD70		R14	
	Return to caller.....>	PDRT			
PD30	No room in current block:				
	Indicate end-of-block in previous record.		DRGP(IPW$DDR)		
	Write data block.....>	PD70		R14	
	Update disk request word.....>	PD40		R14	
	Handle current record.....>	PD08			
	Increment Disk Address/Get Next Track Group Subroutine:				
PD40	Increment disk address subroutine.				
	If disposition field indicates tape spooling, return.....>	PD59			
	Set function track indicator to R.		TCFT(IPW$DTC)		
	Set addressability for DMB.			R6	
	If D-file not on FBA device.....>	PD41			
	Add unit of transfer to current block number in TCB (next D-record).		TCDW(IPW$DTC)		
	Change number of blocks in Q-record.		QRFB(IPW$DQC)		
	If next block will fit into block group, return to caller.....>	PD59			
	Otherwise, get next record.....>	PD52			
PD41	Update seek address record number.		TCDW(IPW$DTC)		
	If not end of track group.....>	PD59			
PD52	Lock DMB.				IPW$RSR
	If no queue record available (MRQF=0)				
	• Issue warning message 1Q38I				IPW$GAM
	• Unpost ECB in DMB		QCEB(IPW$DQC)		

Labels	Chart PD: IPW$$PD - Put Data Record		Modified Data Fields	Reg. Usage	Calls
PD55	If it is save account, set cancel code in TCB and call the terminator.		TCTT(IPW$DTC)		
	• Unlock DMB				IPW$RLR
	• Wait for queue record (posting or ECB)				IPW$WFC
	• Lock DMB				IPW$RSR
	Address free queue record and read it into the auxiliary area.		QCQW(IPW$DQC) QCDS(IPW$DQC)		IPW$RDQ
	Update next-in-set pointer and write current queue record.		QRNS(IPW$DQR)		IPW$WTQ
	Set next-in-set pointer to zero and write new one.		QCNS(IPW$DQR)		IPS$WTQ
	Update statistical information in DMB				
	• Decrement number of available queue records		NRQR(IPW$DQC)		
	• Calculate maximum queue records in use		NRQM(IPW$DQC)		
	Update queue record area:				
	• Set first-in-set switch off		QRFS(IPW$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)		
	Unlock DMB if not save-account task.				IPW$RLR
PD59	Set function track indicator to 0.		TCFT(IPW$DTC)		
	Return to caller.			R14	
	Write Block To Disk/Tape Subroutine				
PD70	Set function track indicator to P.		TCFT(IPW$DTC)		
	If tape spooling in disposition field in queue record, write tape -->	PD77			IPW$WTT
PD75	If not execution processor --------->	PD76			
	If end of data --------------------->	PD76			
	Set double buffer switch.		TCDB(IPW$DTC)		
PD76	Write block to disk.				IPW$WTQ
PD77	Set function track indicator to 0.		TCFT(IPW$DTC)		
	The output block area is reset to zeros if required.		IPW$DDR		
PD78	The remaining block length is reset.		TCBC(IPW$DTC)		
	The previous record pointer is reset to start of block.		TCPR(IPW$DTC)		
	Return to caller.			R14	



IPW$$PF - Mainline

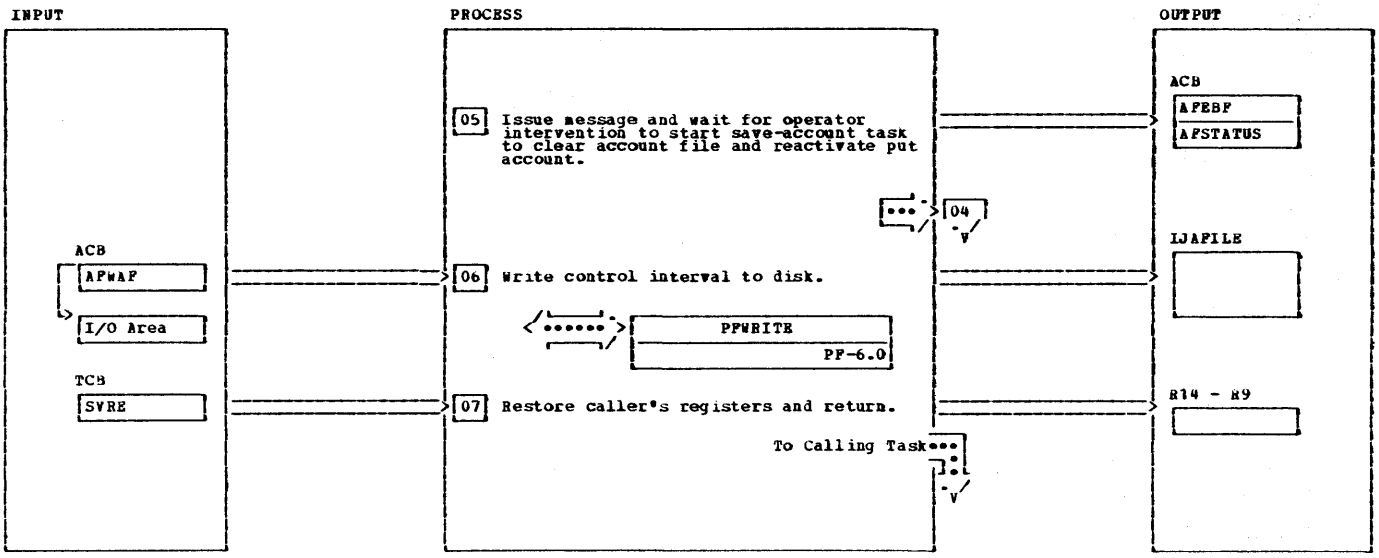


IPW$$PF - Mainline

NOTES	MODULE	LABEL	REF
1 Save caller's registers in caller's TCB The length and address of the account record are passed in R0 and R1 to IPW$$PF by the IPW$PAR macro If the DOS/VSE job-accounting interface is cancelled after VSE/POWER has been initialized, VSE/POWER accounting also is terminated.		PFINIT	$WTO
One of the following messages is issued: 1Q84I ACCOUNT SUPPORT CANCELLED 1Q3AI ERROR WHILE PROCESSING ACCOUNT RECORD RC=xx			$RLR
3 The unit of data transfer between VSE/POWER and the FBA device is a control interval. A control interval may consist of one or more FBA blocks. If no CIDF exists, the control interval is empty. See also listing of PFW$$PF (Prologue) for detailed description of a control interval. If an account record fits into an existing control interval (Step 2), this control interval will be updated and written back to the account file. As one PUT ACCOUNT has exclusive control over the account file, no other task may use the space on disk in the meantime.		PFNOTFIT	

NOTES	MODULE	LABEL	REF
4 As only one task has exclusive control over the account file, other tasks issuing a PUT ACCOUNT request will wait for the locked resource. A SAVE ACCOUNT task may be started by the central operator via the PACCOUNT command to save and/or erase the content of the account file and to clear space to allow processing of the PACCOUNT command to continue.		PFAGAIN	$DPT

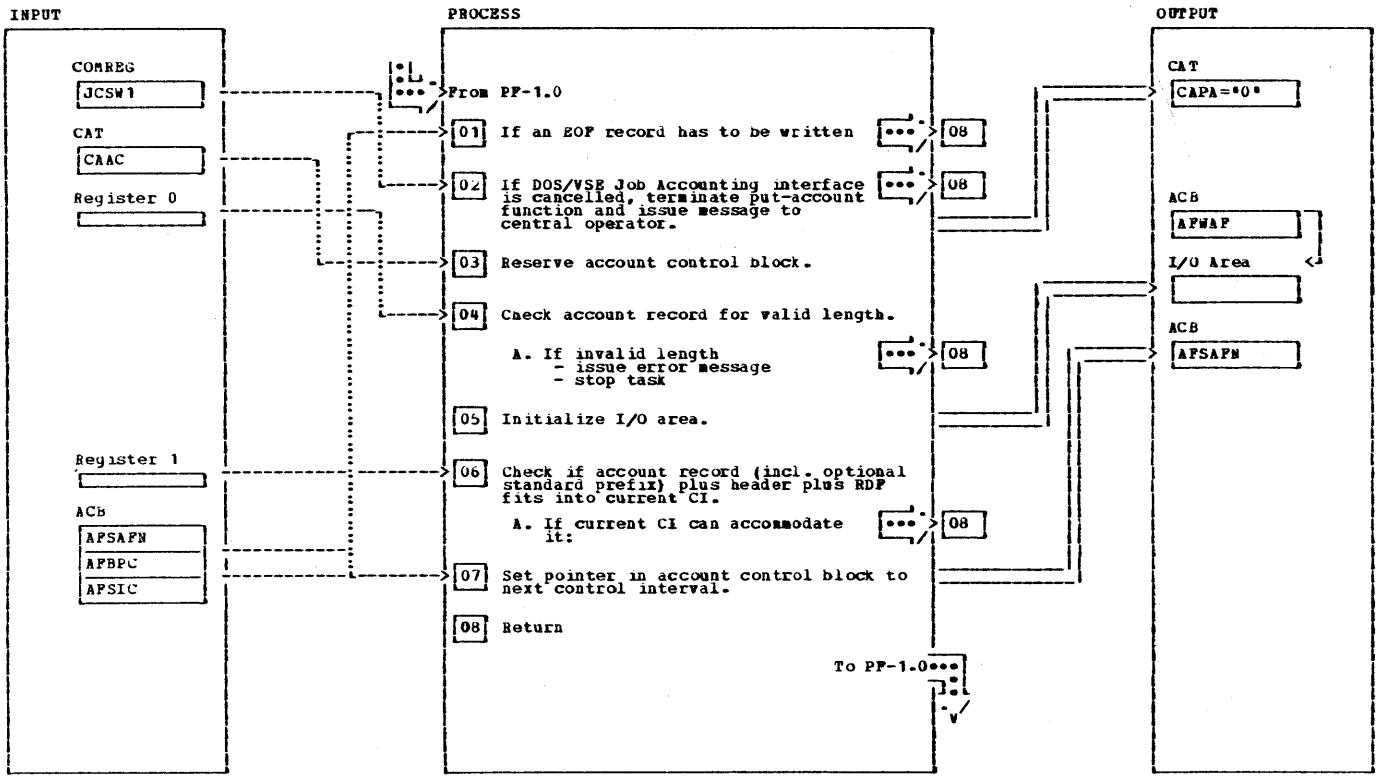
IPW$$PF - Mainline



NOTES	MODULE	LABEL	REF
6 The updated or newly created control interval is written to the account file.			

NOTES	MODULE	LABEL	REF
7 The caller's registers are restored from the save area addressed by register 13, and return is made to him.			$RET

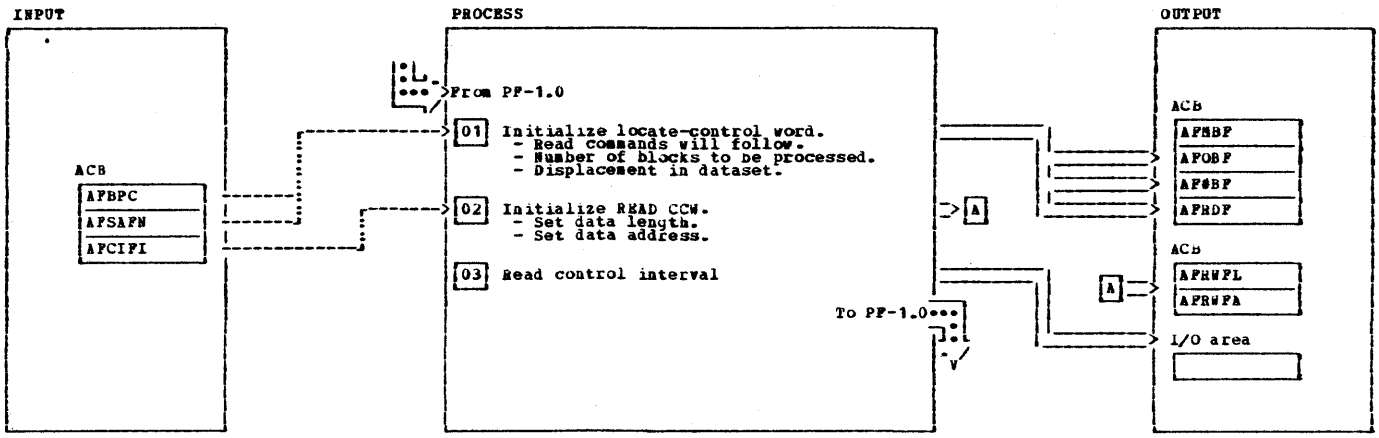
IPW$$PF - PFINIT Initialize for Put Account Request



NOTES	MODULE	LABEL	REF
1 If the length of the account record to be written is zero (R0=0), that means that an EOF record has to be written to the Account file. This has to be done when PEND has been issued to close the Account file properly.			
2 First the 'put-account' indicator is set into the account-function trace byte in the TCB of the calling task to signal the status in case of abnormal end. If 'put account' is called at VSE/POWER initialization time, or if save account is running and spools to the punch queue (which invokes the logical reader, which itself would write unwanted account records), the request is ignored, and return is made to the caller.		PFINIT	
2 If the DOS/VSE Job Accounting interface is cancelled due to an error that occurred in the $JOBACCT module, message 1Q84I ACCOUNT SUPPORT CANCELLED is issued and the VSE/POWER accounting functions are terminated.			$GAM
3 The account control block is locked to exclusively control its content and the account file by this task.			$RSR
4 The length and the address of the account record to be written are passed in register 0 and register 1 respectively. These parameters are saved in the ACB for later usage.			
4 First the length of the account record is checked to see if it exceeds the maximum allowable length. One account record may not exceed the size of a control interval. If the record is too long, message 1Q8AI ERROR WHILE PROCESSING ACCOUNT RECORD, RC=xxx is issued to the central operator, and the task is flagged to stop immediately.			$GAM

NOTES	MODULE	LABEL	REF
5 The I/O area is cleared to binary zeros.			
6 Several account records (preceded by a sequential-file header, and accompanied by a record-description field) may be blocked into a control interval. Account records do not span control interval boundaries. If the remaining space in a CI cannot accommodate an account record, and a new CI is constructed; this remaining space will never be used for other account records.			
6 A check is made to see if the new account record its 8-byte header for block and record length plus a record definition field will fit into the possible free space of the current CI-block. When a SYSID has been specified at VSE/POWER generation, the account record is preceded by a standard prefix, containing the system id , component id (in our case 'SCPWR'), and some other control information such as release and version number.			
7 The current FBA-block pointer is incremented by the number of FBA blocks required for one control interval, to point to the next control interval on the account file, which then will be used in the PFWRITE routine. Return is made to the Mainline to bypass the read function for the current CI.			

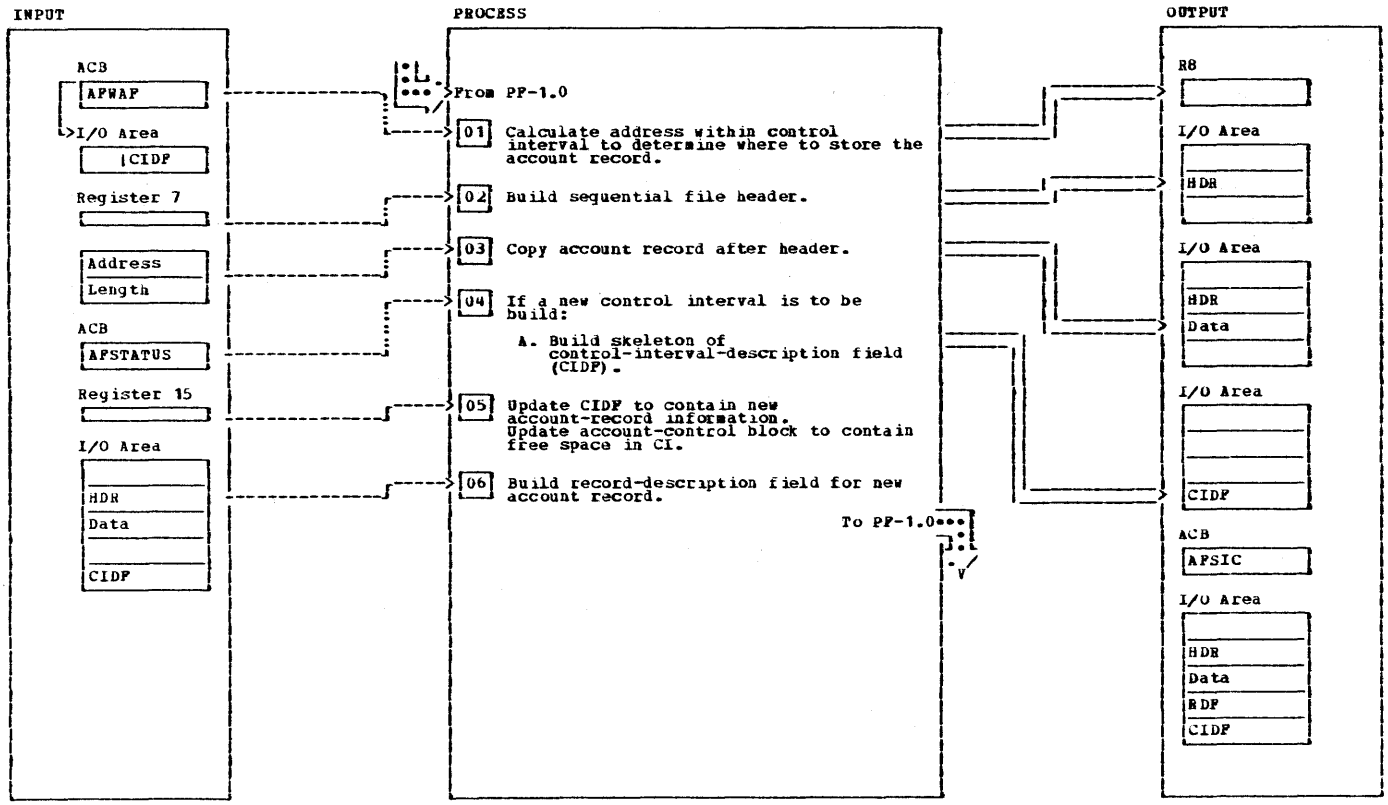
IPW$$PF - PFREAD Read Current Control Interval from Account File



	NOTES	MODULE	LABEL	REF
1	The extent-description block, locate-control word, CCW chain and CCB located in the account-control block are used to read the current control interval from the VSE/POWER account file.		PFREAD	
1	The locate control word is set up with following values: - number of FBA block to be processed - relative block number - indication that read follows			

	NOTES	MODULE	LABEL	REF
2	The define-extent CCW, locate CCW and READ CCW are already chained, and point to their control blocks. The data length set to the READ CCW is the length of one control interval. The address points to the I/O area, containing the control interval.			
3	A DOS/VSE EXCP is issued to read the control interval, and a VSE/POWER WAIT is issued to wait for I/O completion.	EXCP		$WFC

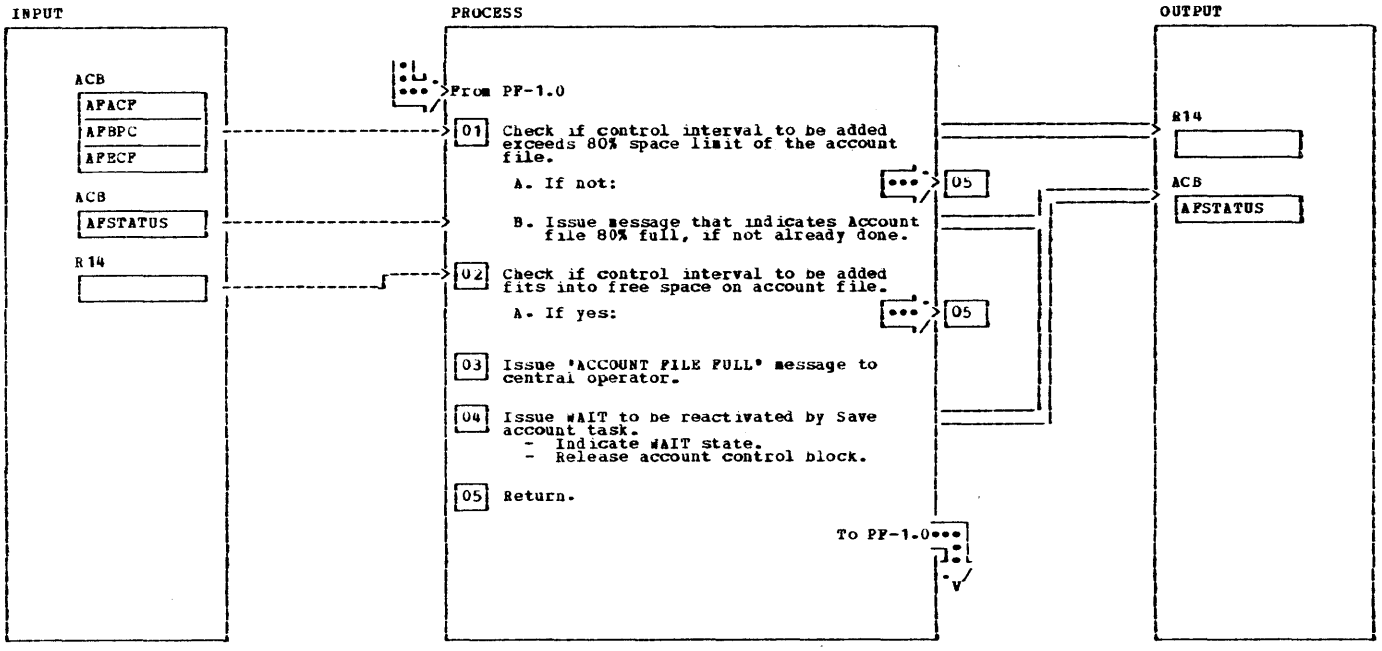
IPW$$PF - PFINSERT Insert Account Record in Current Control Interval



NOTES	MODULE	LABEL	REF
1 PFINIT already has verified that the account record (including the account data), a record description field (RDF), and a sequential file header, fit into the current control interval.		PFINSERT	
2 Layout of the sequential-file header: Byte 0,1: Block length: length of data plus 8 bytes. Byte 2,3: Zeros. Byte 4,5: Record length: length of data plus 4 bytes.			
3 The new account record, addressed by register 14, is copied from the user area after the header in the control interval. When a SYSID has been specified at VSE/POWER generation, the account record is preceded by a standard prefix containing the system id, component id (in our case *SCDPWR*), and some other control information such as release and version number.			

NOTES	MODULE	LABEL	REF
4 It is determined if the current control interval has to be updated with the new account record, or if a new control interval has to be created. The later happens if the new account record does not fit into current control interval or if an EOF record has to be written.			
4 If an existing CI is updated, a CIDF already exists. Else a CIDF skeleton is constructed.			
5 The length of the account record to be inserted contains its data length, a header and an RDF.		PFINSRT1	
6 A scan is made thru the control interval from the end backwards to find free space if RDFs already exist.			

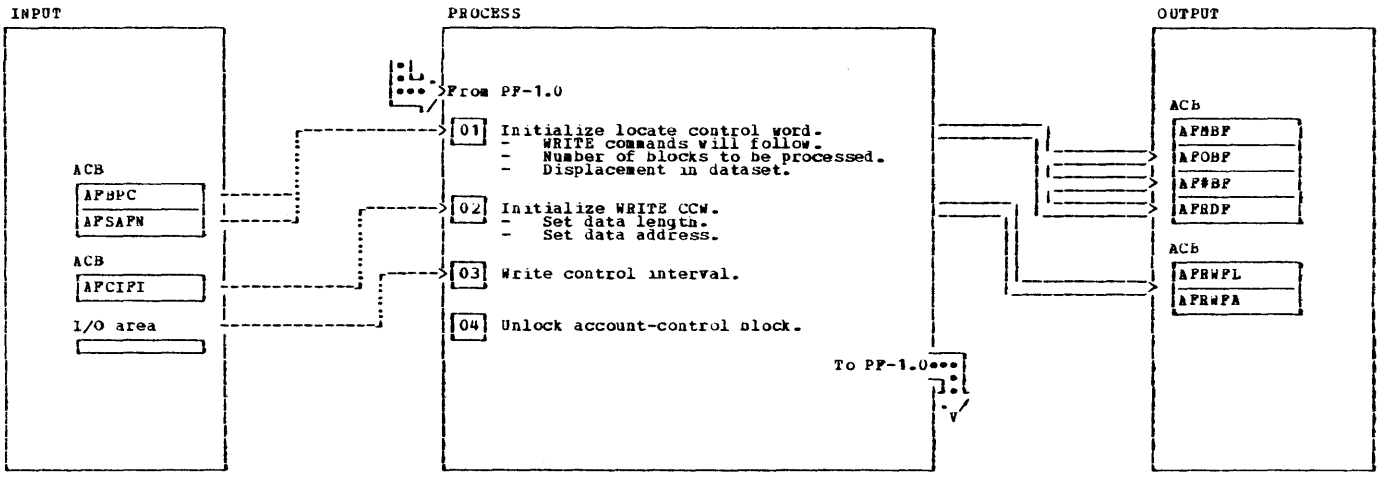
IPW$$PF - PFLIMIT Check if New CI Exceeds Limits on Account File



NOTES	MODULE	LABEL	REF
1 VSE/POWER initialization routines have calculated the number of FBA blocks, which constitute 20% of the overall space in the account file. This value is checked against the remaining free space on the account file. If the last control interval is updated, it will be written back to the Account file, and no check has to be made. Otherwise a check is made to determine if now 80% of all control intervals are in use.		PFLIMIT	
1B The following message is issued the first time when the remaining free space on the account file is less than the 20% space value: 1031I MORE THAN 80% FULL ACCOUNT FILE (IJAFILE)			$GAM
2 No additional action required in this routine, when the newly created control interval can be written to the account file.		PFLIMIT1	

NOTES	MODULE	LABEL	REF
3 The following message is issued to the central operator to inform him to start a save account task, using the PACCOUNT command with its various options: 1032I NO MORE ACCOUNT FILE (IJAFILE) SPACE FOR ttt,cuu The current block number, pointing partially already outside of the Account file extent, is reset to point back to the last used control interval.			$GAM
4 Set put-account task to wait state, to wait for completion of the save-account task, which then will reactivate the put-account task. The account-control block is released from this task. When the task regains control, return is made to the mainline to the beginning of the function.			$WPC

IPW$$PF - PFWRITE Write Control Inverval to Account File



NOTES	MODULE	LABEL	REF
The extent-description block, LOCATE control word, CCW-chain, and CCB located in the account control block are used to write the control interval to the VSE/POWER Account file.		PFWRITE	
1 For each newly created control interval, the number of FBA blocks per control interval is doubled in the LOCATE control word to write the CI containing the account data followed by a CI of same length containing all binary zeros to the account file. So with each added CI an EOF record is written automatically. However this is only done when two or more control intervals fit into the Account file. For updated CIs or if last CI on account file is written, the LOCATE control word contains only information for this CI.			

NOTES	MODULE	LABEL	REF
1 The locate-control word is set up with following values: - number of FBA block to process - relative block number - indication that write command follows			
2 The define-extent CCW, LOCATE CCW, and WRITE CCW are already chained, and point to their control blocks. The data length set to the WRITE CCW is the length of one control interval. The address points to the I/O area containing the control interval.		PFWWRITE1	
3 A DOS/VSE EXCP is issued to write the control interval and a VSE/POWER WAIT is issued to wait for I/O completion.	EXCP		$WPC
4 The account-control block is unlocked from the put account task.			$RLR

IPW$$PL - VSE/POWER Physical List	
Label	Routine
PL00	Function Entry
PL08	End of Job
PLA0	Abnormal Condition Routine
PL10	Printer Command Code Check
PL18	Initialize CCW's and Data
PL32	SETPRT Request Handler
PL40	Print Output Buffer
PL50	Print Output Buffer (double buffering)
PL70	FCB/UCS Processing Routine

Services Used	
Service	Macro
Task Management	IPW$WFC IPW$WFO
Storage Management	IPW$RLW IPW$RSW IPW$RSV
Message Service	IPW$GAM IPW$WTR

Interfaces used	
Module	Macro
IPW$$NU	IPW$OLI - Storage Management
IPW$$LR	IPW$GLR

Functions used	
Module	Macro
IPW$$AS	IPW$IAS
IPW$$LU	IPW$ULP
IPW$$OT	IPW$OTP

Labels	Chart PL: IPW$$PL - Physical List		Modified Data Fields	Reg. Usage	Calls
PLDS	The first 16 bytes constitute the section descriptor:				
	'PLCS release'				
	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$DSV	R12	
	1: device information byte 0 - number of buffers '01' one print buffer '02' two print buffer 'F2' two print buffer and two data buffers byte 1 - device type byte 2-3 - programmer logical unit number			R1	
	<u>Function Entry</u>				
	The physical list routine is entered after the command processor has initialized a list task control block and printer device information (passed in parameter register 1) whenever a PSTART command is given for a list task.				
PL00	The entry parameters in register 1 are saved in register 6.			R6	
	The interface with the logical writer is opened through an IPW$OLI call.				IPW$OLI
	When this is a tape spooling task, branch to initialize and format tape control block.				IPW$OTP
	If task is in stop state............>	PLA9			
PL01	Otherwise, reserve storage for the printer TCB extension area and save address in TCB.		TC3E(IPW$DTE)		IPW$RSV
	If no storage is available and task is in stop state, branch............>	PLA8			

Labels	Chart PL: IPW$$PL - Physical List		Modified Data Fields	Reg. Usage	Calls
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 which will be initialized with the device-dependent printer data and pointers to the printer CCB, CCWs, and data records to be printed by this task.				IPW$RSW
	Physical work space is made addressable through register 8.		IPW$DPW	R8	
	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$RSW
	The appropriate PDA size is obtained from the permanent area.		BLBF(IPW$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.		PBV1(IPW$DPW) PBR1(IPW$DPW)		
	Branch to continue..... >	PL06			
PL05	The second PDA is reserved for CCB, related CCWs and data buffers.			R1,R2	IPW$RSW
	The virtual and real addresses of the CCB (first 16 bytes of the PDA) are saved in the PWS.		PBV2(IPW$DPW) PBR2(IPW$DPW)		
PL06	Now the first 16 bytes of the buffer space are set up as a CCB, indicating:				
	• Wait for device end		CBC1(IPW$DCB)		
	• Accept unrecoverable I/O error		CBC1(IPW$DCB)		
	• Command chain retry		CBC2(IPW$DCB)		
	• Printer logical unit number		CBLC(IPW$DCB)		
	• Set EXCP real.		CBLC(IPW$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 PL: 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 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	
	Get FCB phase name suffix for PRT1 type printer..... >	PL50		R14	
	Get 3800 model type.....>	PL510			
	Branch to continue..... >	PL10			

Labels	Chart PL: 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$RLW
	If a second PDA exists, it is released too.			R1	IPW$RLW
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
	Register 8 is set to zero to show that the PWS is released			R8	
	Branch to continue.....	PL02			

Labels	Chart PL: IPW$$PL - Physical List		Modified Data Fields	Reg. Usage	Calls
	Abnormal Condition Handler				
	This routine is entered whenever an unrecoverable I/O error has been detected by DOS/VSE.				
PLA0	Prepare reply area. Issue message 1Q61D and wait for reply.		PWRA(IPW$DPW) PWML(IPW$DPW)	R5,R6	IPW$GAM IPW$WTR
	If reply is 'C', branch to..... >	PLA8			
	If reply is 'I' and single buffering is being used, branch to..... >	PL44			
	If reply is 'I' and double buffering is being used, branch to..... >	PL54			
PLA2	If reply is not 'R', branch to..... >	PLA0			
	If a backup count is specified, branch to..... >	PLB0			
	Branch to count skips to channel one not yet printed..... >	PLC0		R14	
	Set up for restart number of pages + 1.		TCRS(IPW$DTC)	R5	
PLA6	If single buffering is being used, branch to..... >	PL48			
	Clear print buffers. Branch to.... >	PL66		R1,R2 R14	
	Indicate new start.		PWVE(IPW$DPW)		
	Branch to get next record..... >	PL26			
PLA8	Set termination byte to 'S' (Stop). Branch to termination routine.		TCTT(IPW$DTC)	R2	IPW$$TR
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$DTC)	R1,R2 R5	
	Branch to count number of skip to channel ones not yet printed..... >	PLC0		R14	
	Add number of skip to channel ones not yet printed to number of pages to go back (specified by operator).		TCRS(IPW$DTC)	R5,R6	
	Branch to..... >	PLA6			
	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	

Labels	Chart PL: IPW$$PL - Physical List		Modified Data Fields	Reg. Usage	Calls
PLC2	(Note: NOP-CCW without chain-flag stops CCW-chain).			R5,R6	
	The remaining CCW-chain is scanned for skip to channel one operation code. If so, branch to..... >	PLC4			
	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.			R6 R5	
	Return to the caller..... >	R14			
	Phase Load Subroutine:				
PLL0	Build directory entry to load phase:				
	• Insert phase name into skeleton directory entry			R1,R2	
	Fill in the parameter list for the FETCH routine. The load list is placed behind the directory entry in the work area.			R0,R1 R2	
	• Store the phase name address • Store the load list pointer • Set option switch to indicate no text loading.		LLNAME(IPW$DEF) LLDEADR(IPW$DEF LLNTXT(IPW$DEF)		
	Load phase by issuing SVC 4 If the phase is not present in the core image library, or if its size does not fit in the supplied work area return to caller via register 14 (error exit)..... >	R14		R0,R1	SVC 4
	Save the length of the phase.			R15	
	Set the option switch in the parameter list to indicate loading of text and issue SVC 4.			R0,R1	SVC 4
	Return to caller..... >	R14		R15	

Labels	Chart PL: IPW$$PL - Physical List		Modified Data Fields	Reg. Usage	Calls
	Insert Device Dependent FCB Prefix				
	This subroutine inserts the proper prefix in the FCB phase name when the first four characters of the specified FCB name are '$$$$'. The prefixes inserted are: FCB1 -- 3800 FCB2 -- PRT1 family FCB3 -- 3203 FCB4 -- 5203				
PLF0	When the generic FCB phase name is specified, plug in proper prefix.			R1	
	Return to caller..... >	R14			
	Get Sense Information				
	This subroutine issues a sense I/O command when the printer being used belongs to the PRT1 family. The standard FCB phase name suffix is obtained from the sense information.				
PLS0	If printer is not PRT1 type printer, return to caller ..... >	R14			
	Otherwise, construct sense CCW. Place sense area just behind sense CCW in PDA.		CWCC (IPW$DCW) CWDA (IPW$DCW)	R1, R2	
	Do I/O and wait for its completion.			R1	IPW$WFC
	If I/O error occurred, set stop code in TCB and branch to termination routine..... >	PLA9	TCTT (IPW$DTC)		
PLS2	Register 6 is reloaded with the address of the first CCW. The first CCW is reinitialized as a NOP-CCW with the real address of the end of the PDA.		CWDA (IPW$DCW)		
	Convert FCB name suffix obtained from the sense information, when printer is not a 3211. (Note: The standard suffix for the 3211 printer is ' ').		PWFS (IPW$DPW)		
	Return to caller..... >	R14			

Labels	Chart PL: IPW$$PL - Physical List		Modified Data Fields	Reg. Usage	Calls
	Get 3800 Model Information				
	This subroutine issues a sense I/O command when the printer being used belongs to the 3800 family. A 3800 model flag is set depending on the obtained sense information.				
PLSIO	If printer is not a 3800 printer, return to caller ..... >	R14			
	Otherwise, construct sense I/O CCW. Place sense area just behind sense CCW in PDA.		CWCC (IPW$DCW) CWDA (IPW$DCW)	R1, R2	
	Do I/O and wait for its completion.			R1	IPW$WFC
	If I/O error occurred, set stop code in TCB and branch to termination routine..... >	PLA9	TCTT (IPW$DTC)		
PLSIO2	Register 6 is reloaded with the address of the first CCW. The first CCW is reinitialized as a NOP-CCW with the real address of the end of the PDA.		CWDA (IPW$DCW)		
	If printer is an ISP model of the 3800, set appropriate flag in printer TCB extension area.		PTEFLAG (IPW$DTE)		
	Return to caller..... >	R14			

Labels	Chart PL: 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..... >	PL19			
	If valid write command, ...>	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..... >	PL19			
	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	00: Invalid, branch to ignore command..... >	PL25			
	04: Valid, branch to handle record..... >	PL19			
	08: Branch to empty buffer first.. >	PL16			
	0C: Branch to perform FCB/UCS load >	PL70			
	10: Branch to perform printer setup..... >	PL32			
PL16	A link is made to print PDA..... >	PL40		R14	
	Branch to continue record handling..... >	PL18			

Labels	Chart PL: IPW$$PL - Physical List		Modified Data Fields	Reg. Usage	Calls
	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$GLR call is issued to logical writer to get the next data record. The CCW string is started right behind the CCB at the beginning of the data buffer space and the data records are chained backwards in the PDA, starting at the end of the PDA. This procedure is followed to optimize the use of PDA space, since the variable length list records prevent any calculation of a data record start address if data records are chained forward normally in the PDA. If insufficient PDA is available, the data buffer is emptied first by writing the data records to the printer.				
PL18	If the printer is a 3800 and the 1st byte of the record is the TRC byte (OPTCD=J), the appropriate CCW is built.				
	If printer is not a 3800.....>	PL19			
	If no TRC processing is requested...>	PL19			
	Strip off TRC byte and adjust record length			R3, R4	
	Isolate new TRC indicator and if corresponding CCW is already issued >	PL19			
	Check if select translate table CCW fits into current buffer. If yes....>	PL18B			
	Otherwise, a link is made to empty buffer.....>	PL40			
PL18B	Set up select translate table CCW and chain NOP-CCW to it.....>	PL94	CWCC(IPW$DCW)		
PL19	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			

Labels	Chart PL: IPW$$PL - Physical List		Modified Data Fields	Reg. Usage	Calls
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			
PL22	The command code, contained in the high-order byte of register 3, is stored in the new CCW.		CWCC(IPW$DCW)		
	The real data address, now pointing to the end of the available data buffer space, is loaded in register 2, the record length contained in register 4 is subtracted, and the updated real data address and data length are stored into the CCW.		CWDA(IPW$DCW) CWCT(IPW$DCW)	R2	
	When channel one has been crossed already, a branch is made to move data record to printer I/O-buffer.. >	PL23			
	Otherwise command code is set to NOP, also New Page Flag is set.		CWCC(IPW$DCW) CWRE(IPW$DCW)		
PL23	Register 6 is set to point to the next CCW, which is now made the NOP CCW terminating string, while its data address is made to point to the end of the available data buffer space after moving the current data record by branch and link .....>	PL94	CWDA(IPW$DCW)	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..... >	PL24			
	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..... >	PL2A			

Labels	Chart PL: IPW$$PL - Physical List		Modified Data Fields	Reg. Usage	Calls
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	
PL2A	If the CCW operation code was a clear printer X'87', a branch is made to empty the print buffer(s)..... >	PL40			
	If double buffering, branch to wait for the completion of the last I/O. >	PL50	PWOT(IPW$DPW)	R14	
	(This is necessary because the double print buffer processing routine does not wait for the completion of the I/O.)				
	Continue mainline .....>	PL26			
PL25	Check if the last processed data record is the last record in the data buffer. If not so, or if double buffering is being used, or if the PDA is empty, branch to..... >	PL26			
	Load the CCB address into register 1, and execute the channel program via SVC 0.			R1	
	Request the next record from the logical writer..... >	PL30			
	Wait for I/O completion..... >	PL41		R14	
	Branch to..... >	PL27			
PL26	The next data record is requested from the logical writer through an IPW$GLR call..... >	PL30		R2	
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$DPW)	R14	
	Branch to End of Job routine..... >	PL08			
PL30	Request the next record from the logical writer.				IPW$GLR
	Save record address and length, respectively, in register 3 and register 4, and return to caller.			R2 R3,R4	

Labels	Chart PL: IPW$$PL - Physical List		Modified Data Fields	Reg. Usage	Calls
	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		R14	
	If double buffering is being used, a wait request is issued by branching to..... >	PL50	PWOT(IPW$DPW)	R14	
	(This is necessary because the double print buffer processing routine does not wait for the completion of the I/O.)				
PL34	When the DUMP/TRACE option has been specified in the SETPRT parameter list, SYSLST is assigned to the same printer being used.				IPW$ULP
	If SYSLST is already assigned, message 1R64I is written.				IPW$GAM
	The dump/trace flag is turned off.		PTEFLAG2 (IPW$DTE)		
PL36	Storage for the service request block (SRB) is acquired.				IPW$RSW
	The SRB is formatted:				
	• SETPRT request		SRBREQ (IPW$DSR)		
	• Address of SETPRT parameter list		SRBPARM (IPW$DSR)		
	The TCB extension area is made addressable using register 2 as base.			R2	
	The proper programmer logical unit is inserted in the SETPRT parameter list.		SPPLUSYS (SPLIST)		
	The actual copy group index is inserted in the parameter list.		SPPCINDX (SPLIST)		
	The operator message suppress flag is set.		SPPFLAG1 (SPLIST)		
	If setup is requested, branch to... >	PL37			
	Otherwise, check if the new SETPRT request matches the previous request. If so, skip setup processing.....>	PL38			
	Copy the SETPRT parameter list into the TCB extension area.		PTELIST (IPW$DTE)		

Labels	Chart PL: IPW$$PL - Physical List		Modified Data Fields	Reg. Usage	Calls
PL37	The service request block is passed to asynchronous service for processing by means of the IPW$IAS TYPE=SERVICE macro instruction.				IPW$IAS
	Turn off flags for FCB verification, mark form, and offset stacking.		PTEFLAG2 PTEFLAG1		
PL38	Turn off setup required flag				
	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
	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.				
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	
PL41	On return, an IPW$WFC call is issued to have the list task wait for I/O completion.				IPW$WFC
	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			

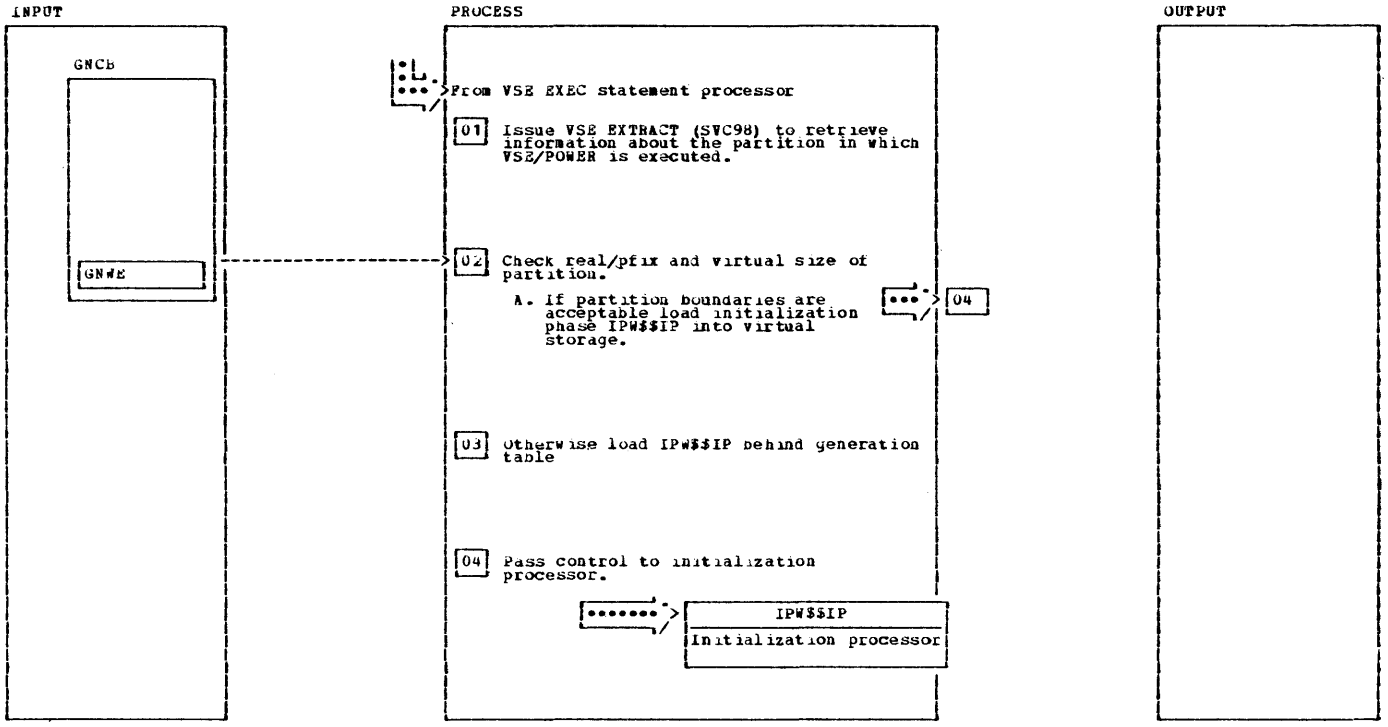
Labels	Chart PL: IPW$$PL - Physical List		Modified Data Fields	Reg. Usage	Calls
PL44	Otherwise, the CCW chain interruption address is made CCW start address in the CCB, and a branch is made to the subroutine entry to restart the I/O at the point of interruption..... >	PL40	CBCA(IPW$DCB)		
PL48	The CCW start address in the CCB, which might have been changed on processing a channel 9 or 12 overflow, is now reset using the address saved in the PWS, and register 6 is set to point to the first CCW again.		CBCA(IPW$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$WFC
PL52	The CCB is checked for unrecoverable I/O error. If so, branch to ask operator for proper action..... >	PLA0			
	The CCB is checked for unit exception (channel 12 overflow), and ignored errors.				
	If so, a branch is made to restart I/O to the printer at the point of CCW chain interruption..... >	PL54			
	If channel 9 overflow is indicated in the CCB, a branch is made to bypass I/O restart..... >	PL58			
PL54	Otherwise, the CCW chain interruption address is made CCW start address in the CCB. An SVC 0 is issued. Branch to wait for I/O completion.. >	PL51	CBCA(IPW$DCB)		

Labels	Chart PL: IPW$$PL - Physical List		Modified Data Fields	Reg. Usage	Calls
PL58	Registers 1 and 2 are loaded with the virtual and real addresses of the other print buffers respectively.			R1,R2	
PL60	If this is not a wait only request -> Otherwise set virtual address of active buffer to zero to indicate first time processing	PL61	PWVE(IPW$DPW)		
	Load registers 1 and 2 with the virtual and real addresses of first buffer			R1,R2	
	Turn off wait only flag, bypass SVC0 processing ----------->	PL66	PWOT(IPW$DPW)		
PL61	Registers 1 and 2 are stored in the PWS to address the active print buffer.		PWVE(IPW$DPW)		
	Register 2 is loaded with the real address of the first CCW in the PDA, which is stored in the CCB and in the PWS.		CBDA(IPW$DCB) PWCA(IPW$DPW)	R2	
	Register 1 contains the virtual address of the CCB as stored in the active print buffer and serves as parameter register for the EXCP (SVC 0), which is now issued.				
PL62	Registers 1 and 2 are loaded with the virtual and real addresses of the available print buffer.			R1,R2	
PL66	The addresses of the real and virtual end of the buffer space are stored in the PWS.		PWDA(IPW$DPW) PWDV(IPW$DPW)	R5	
	Register 6 is set to point to the first CCW again.			R6	
	The first CCW is initialized as a NOP-CCW and the associated data area address is initialized with the real address of the end of the PDA.		CWDA(IPW$DCW)		
	A branch is made back to the caller.			R14	
	FCB/UCS Processing				
PL70	Check if printer being used is a FCB type printer. If not, branch to... >	PL80		R1	
PL74	Check if a FCB name is specified. If not, take DOS/VSE default FCB name. The default FCB names are: 3211 -- $$BFCB 3202 -- $$BFCB3 5203 -- $$BFCB5 3203/4 -- $$BFCB00 3289-E -- $$BFCB10		NDHGFCB (IPW$DNR)		

Labels	Chart PL: IPW$$PL - Physical List		Modified Data Fields	Reg. Usage	Calls
PL76	Insert standard FCB prefix, when a generic FCB name is specified..... >	PLF0	NDHGFCB (IPW$DNR)	R14	
	If the FCB to be loaded is the same as the one on the printer, branch ..>	PL80		R1	
	Save the new FCB name and reset the current forms id to force the mount forms message to be isssued.		PTEFCBN LWFI(IPW$DTC)		
	If the print buffer is not empty ...>	PL77			
	Setup a dummy I/O (write no-space) to ensure that the printer is ready, and chain it into the CCW, ...>	PL94			
PL77	Empty the print buffer(s)..... >	PL40		R5,R14	
	If two print buffers are used, branch to wait for the completion of the last I/O..... >	PL50	PWOT(IPW$DPW)	R14	
	The general workarea of the printer TCB extension area is used to build the LFCB parameter list.				
	• Convert logical unit number from CCB into LFCB parameter list.			R0,R1 R2	
	• Insert FCB phase name.				
	Storage for the service request block (SRB) is acquired and formatted.				IPW$RSW
	• LFCB request		SRBREQ(IPW$DSR)		
	• Address of LFCB parameter list		SRBPARM		
	• Address of phase name		(IPW$DSR)		
	Call asynchronous service to process the request.				IPW$IAS
	The service request block is released and returned to the storage pool.				IPW$RLW
	If load FCB was successful, branch to UCS processing..... >	PL80		R15	
	Otherwise branch to..... >	PL26	TCTT(IPW$DTC)		
PL80	Check if UCS load is requested. If not, branch to get next record..... >	PL26		R1	
PL81	Check if the printer being used is a UCS type printer. If not, branch... >	PL26		R1	
	If the UCS buffer is already loaded, done by a previous job, branch to.. >	PL26			
	Empty the print buffer(s)..... >	PL40		R14	
	If double buffering is being used, branch to wait for completion of the last I/O..... >	PL50	PWOT (IPW$DPW)	R14	
PL83	Reserve temporary workspace to load the UCS buffer. (workspace size = DBLK size).			R15	IPW$RSW
	Branch to load UCS buffer into work-space..... >	PLL0		R14	
	Check if the length of the just loaded UCS buffer corresponds with the length expected by the appro-priate printer. If not, branch to.. >	PL88			

Labels	Chart PL: IPW$$PL - Physical List		Modified Data Fields	Reg. Usage	Calls
	Save new UCS phase name and associated options.		PTEUCSN (IPW$DTE)		
	Tell operator to mount proper print-train and wait for his reply.				IPW$GAM IPW$WFO
	When he answers STOP, FLUSH or FLUSH HOLD, branch to..... >	PL89			
	Construct CCW chain for UCS buffer loading: • Build allow/block datacheck CCW depending what is specified • Build fold/unfold CCW • Build UCS buffer load CCW and move UCS buffer from work area to print buffer.				
PL87D	Release temporary workspace.				IPW$RLW
	Branch to get next data record..... >	PL26			
PL88	Inform operator that a UCS error occurred.				IPW$GAM
	Set immediate-stop code in TCB and branch to..... >	PL26	TCTT(IPW$DTC)		
PL89	Reset current UCS name and branch to..... >	PL87D	PTEUCSN (IPW$DTE)		

IPW$POW VSE/POWER LOader



NOTES	MODULE	LABEL	REF
The Loader routine is in front of the generation table. The source code is in the POWER macro. The phase name is given by the user. There can be as many of these phases in the CIL as there are different versions of VSE/POWER needed by the user.			
1 The EXTRACT macro uses a work area which is part of the generation table (GNCB)	EXTRACT	IPW$POWI	

NOTES	MODULE	LABEL	REF
2 The partition boundaries are not acceptable if the virtual partition size (ALLOC-ALLOCH) is too small to get the initialization processor loaded (10K bytes).			
3 Now IPW$$I1 displays an information message to the central operator and terminates initialization.	LOAD	IPW$POWG	
4 Base register for IPW$$IP is set up and a branch is made to the entry point.	load		

IPW$$PP - VSE/POWER Physical Punch	
Label	Routine
PP00	Function Entry
PP10	Punch Command Code Check
PP30	Initialize CCw's and Data
----	End of Job (Queue entry)
PP60	Punch Output Buffer
PP80	I/O Error Handler

Services Used	
Service	Macro
Task Management	IPW$WFC
Storage Management	IPW$RLW IPW$RSW
Message Service	IPW$GAM IPW$WTR

Interfaces used	
Module	Macro
IPW$$NU	IPW$OLI - Storage Management
IPW$$LR	IPw$GLR

Functions used	
Module	Macro
IPW$$OT	IPW$OTP

Labels	Chart PP: IPW$$PP - Physical Punch	Modified Data Fields	Reg. Usage	Calls
PPCS	The first 16 bytes constitute the section descriptor: 'PPCS release'			
	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$DSV	R13	
	1: device type and logical unit number of punch.		R1	
	Function Entry			
PP00	The physical punch routine is entered after the command processor has initialized a punch task control block and punch device information (passed in parameter register 1) whenever a PSTART command is given for a punch task.			
	The entry parameters in register 1 are saved in register 6.		R6	
PP02	The interface with the logical writer is opened through an IPW$OLI call.			IPW$OLI
	Check if input medium is a tape, rather than queue. If so, branch to..... >	PP04		
	Otherwise, request creation of tape control block.			IPW$OTP
PP04	The first data record is requested from the logical writer through an IPW$GLR call.			IPW$GLR
	On return from the logical writer, register 0 should contain the address of the first data record passed, and		R6	
	register 1 its length.		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.			IPW$RSW
	Physical work space is made addressable through register 8.		R8	
	Device dependent information is stored.	PWDI(IPW$DPW)		

Labels	Chart PP: 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$RSw
	Real address of PDA is saved in register 2.		R2	
	Virtual address of PDA is stored in PWS.	PBV1(IPW$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$DCW)		
	Register 5 is now set up with the virtual address of the end of the PDA, which is then stored in the PWS.	PWDV(IPW$DPW)	R5	

Labels	Chart PP: 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.				
PP10	If command code is X'FF', indicating internal VSE/POWER control record, branch to ignore command..... >	PP48			
	If command is X'X9' and device is 5240P, 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	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. The record length is set to 1.		PPEB(IPW$DTC)		
	If separator cards are requested, the default stacker must be selected. The device type is checked and a branch made to the appropriate routine.				
PP20	Check if separator cards are requested. If not, branch to..... >	PP30			
	Check the device type:				
	If 2560, branch to..... >	PP26			
	If 1442N2, branch to..... >	PP24			
	If 5425, branch to..... >	PP28			
	Otherwise, it is 3525, 2520, or 2540.				
	Check whether it is a write card; if so, branch to..... >	PP30			
	Turn off bits 0 and 1 of the command code.		TCCC(IPW$DTC)		
	Turn on bit 1 of the command code and branch to..... >	PP30	TCCC(IPW$DTC)		
PP24	Turn off bit 1 of the command code, and branch to..... >	PP30	TCCC(IPW$DTC)		
PP26	Check whether it is a load print buffer or write a card. Branch .... >	PP30			

Labels	Chart PP: IPW$$PP - Physical Punch		Modified Data Fields	Reg. Usage	Calls
PP28	Turn on bit 0 of the command code.		TCCC(IPW$DTC)		
	Check for stacker select command. If not, branch to..... >	PP30			
	Turn off bit 2 and 3 of stacker select command, but turn on bit 0 and 1 (stacker select 4) for a punch device other than a 5425; turn on also bit 3 (stacker select 5).		TCCC(IPW$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 PP: 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$DCW)		
	The record length and general purpose byte are saved in the CCW.		CWRE(IPW$DCW) CWCT(IPW$DCW)		
	Register 6 is set to point to the next CCW, which is now made the NOP CCW terminating the CCW string, while its data address is made to point at the end of the available data buffer space after moving the current data record.		CWDS(IPW$DCW) CWDA(IPW$DCW)	R6	
	Register 5 is made to point to the target address of the current data record in the PDA, by subtracting the record length, as contained in register 4, from the address of the end of available buffer space.			R5	
	If the record length is not more than 256 bytes, a branch is made to move the record with a normal MVC instruction, subject of an EXECUTE instruction..... >	PP44			
	Otherwise, registers 2, 1, and 0 are set up as from-address, to-length and to-address, respectively, to be used as the operands of the MVCL instruction that moves the data record.			R2,R1 R0	
	After moving the record to the PDA, a branch is made to obtain the next record from the logical writer..... >	PP46			
PP44	Register 4 is decremented by 1 to contain the record length in machine format and the record is moved using an EXECUTE instruction.			R4	
PP46	A test is made to see whether the PDA has to be emptied. If not, branch to get the next record..... >	PP48			
	Otherwise, the switch in the TCB is reset to X'00', and a link is made to empty the block..... >	PP60	PPEB(IPW$DTC)		

Labels	Chart PP: 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
	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, a link is made to punch the buffer..... >	PP60		R3 R14	
	On return, the physical data area and the physical work space are released and return is made to the logical writer..... >	PP04			IPW$RLW
	Otherwise, a branch is made to process the new record..... >	PP10			
	Punch Routine This routine handles the actual punching of punch records.				
PP60	On entry, register 5, supposed to point to the end of the free buffer space, is compared to the virtual end of data space address in the PWS.				
	If equal, indicating an empty block, immediate return is made to the caller.			R14	
	Otherwise, register 1 is loaded with the virtual CCB address as stored in the PWS, to serve as the parameter register for the EXCP (SVC 0), which is now issued.			R1	
	On return, an IPW$WFC call is issued to have the punch task wait for I/O completion.				IPW$WFC
	Test for unrecoverable I/O error. If so, branch..... >	PP80			
	If not ignored errors, branch to... >	PP64			
PP62	Otherwise, branch to restart from the broken CCW and restart I/O..... >	PP60			

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

IPW$$PR - VSE/POWER Physical Reader	
Label	Routine
PRCS	Entry Processing
PR08	Initialization of physcial data area (PDA)
PR20	Read Data Block Routine
PR30	Deblocking Routine
PR60	EOF/EOJ Processing
PR70	Exit Processing

Services Used	
Service	Macro
Task Management	IPW$WFC IPW$WFI
Storage Management	IPw$RLW IPW$RSW
Message Service	IPw$GAM

Interfaces used	
Module	Macro
IPW$$LR	IPW$PLR - Put Logical Record IPW$OLI - Open Logical Interface IPW$CLI - Close Logical Interface

Called By	
Module	Description
IPw$$CS	PSTART command processor

Labels	Chart PR: IPW$$PR - Physical Reader		Modified Data Fields	Reg. Usage	Calls
PRSD	The first 16 bytes constitute the section descriptor:				
	'PRCS release'				
	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$DSV	R13	
	Function Entry				
	The entry parameters in register 1 are saved in register 6.			R6	
	The entry parameters in register 2 are saved in register 7.			R7	
PR02	The logical reader interface is opened through an IPW$OLI call.				IPW$OLI
PR03	Reserve work space for the reader PWS (physical work space), to save device dependent data for a card reader.				IPW$RSW
	Addressability of PWS is set through register 8.			R8	
	If the reader task has not been started with a connected 3540 diskette reader, (alternate diskette process), branch to.....>	PR05			
PR04	Reserve work space for a 3540 PWS (physical work space), to save device dependent data for a 3540 diskette.				IPW$RSW
	Save the virt address of the PWS in the TCB.		TC3W(IPW$DTC)		
	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 3540 PWS is present and that a alternate diskette process has to be perrormed (LWER=X'01').		LWER(IPW$DTC)		

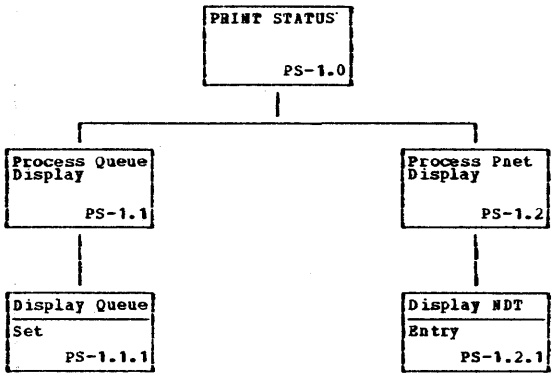
Labels	Chart PR: IPW$$PR - Physical Reader		Modified Data Fields	Reg. Usage	Calls
PR05	Device dependent data is saved in PWS.		PWDI(IPW$DPW)		
	The device type is translated into a displacement (in number of entries) in the DVB (Device Control Block Table) and multiplied by the entry length to get the proper byte displacement in register 7.		PWDT(IPW$DPW) DVDT(IPW$$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
	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$RSW
	The second PDA is loaded from the permanent area.				
	The virtual and real address of the CCB (first 16 bytes of PDA) is saved in PWS.		PVB2(IPW$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)		

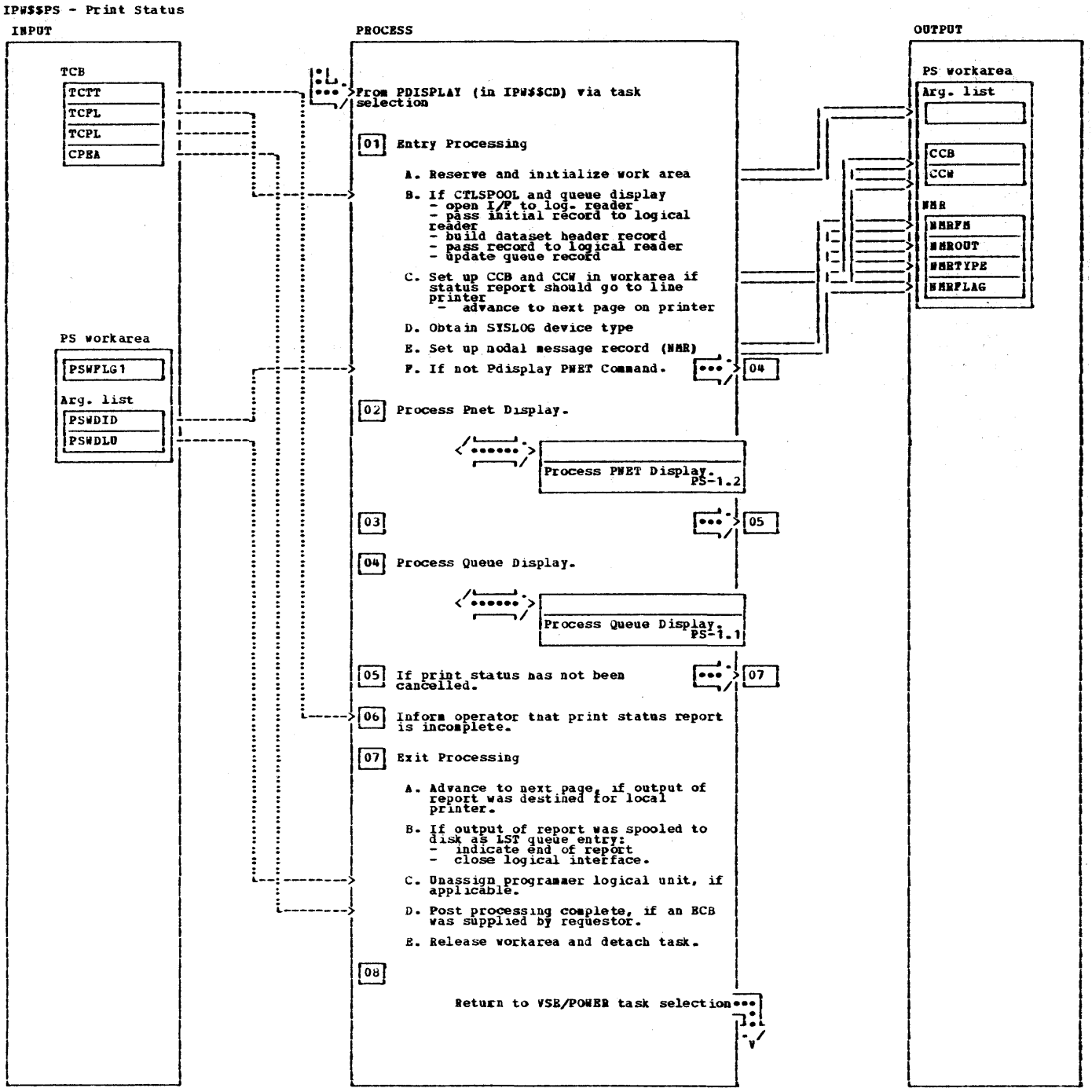
Labels	Chart PR: IPW$$PR - Physical Reader		Modified Data Fields	Reg. Usage	Calls
PR10	The CCW string is set up:				
	A CCW image is moved from the DVB entry to the PDA.		IPW$DCW	R3	
	The real data buffer address contained in register 6 is stored in the CCW.		CWDA(IPW$DCW)	R6	
	Register 4 is incremented 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$DPW)	R1 R2	
	Wait for I/O completion of active buffer. Registers 0, 2, 3, and 6 are initialized for the deblock routine:				
	0: Virtual address first data buffer 2: Real address first CCW 3: Real address CCW last executed 6: Real address last CCW in string			R0 R2 R3 R6	
	If single buffering, branch to get record length..... >	PR28	PWDB(IPW$DPW)		
	If unit exception occurred, branch to get record length..... >	PR28	CBSD(IPW$DCB)		
	Otherwise, issue an SVC 0 for the inactive buffer.				

Labels	Chart PR: IPW$$PR - Physical Reader		Modified Data Fields	Reg. Usage	Calls
PR28	Get record length in register 1, and enter the deblock routine (PR30) at PR40..... >	PR40	PWRL(IPW$DPW)	R1	
	Deblocking Routine				
PR30	Register 0 (virtual address data record) and register 1 (record length) are restored from task save area on return from logical reader.			R0 R1	
	Virtual address data record (register 0) and real address associated CCW (register 2) are incremented.			R0 R2	
PR40	(This entry point is taken whenever a new data block has been read in.)				
	Depending on various conditions, a record is passed to the logical reader or other action is taken.				
	If the record was not the last one read, branch to pass to the logical reader..... >	PR50			
	If last CCW executed was the last CCW in the string, branch to read next block..... >	PR20			
	If unit exception is detected, branch to indicate unit exception..... >	PR42		R4	
	If ignored error, branch to read the next block..... >	PR20			
	Otherwise, pass the last record.... >	PR50			
PR42	If unit exception and device not 2560, branch to indicate unit exception to logical reader..... >	PR45			
	If device 2560, issue an IPw$PLR call to pass last record and issue 2 dummy reads to empty the card path.				IPw$PLR
	Reset DEVICE END switch.		TCG2(IPw$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	

Labels	Chart PR: IPW$$PR - Physical Reader		Modified Data Fields	Reg. Usage	Calls
PR50	Control is passed to the logical reader through an IPW$PLR call.				IPW$PLR
	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	
PR62	The PDA(s) is (are) released.				IPW$RLW
	Register 1 is loaded with PWS address, and PWS is released.			R1	IPW$RLW
	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			
PR64	The physical rdr PWS is released and indicated. R8 is set to zero.		R8		IPW$RLW
	Check If 3540 PWS present, if not...>	PR66			
	Otherwise the PWS will be released and indicated. TCB field set to zero.		TC3W		IPW$RLW
PR66	If unit exception and EOJ, message 1Q34I is issued.				IPW$GAM
	Otherwise check for device end.....>	PR85			

Labels	Chart PR: IPW$$PR - Physical Reader		Modified Data Fields	Reg. Usage	Calls
PR70	The logical reader interface is closed.				IPW$CLI
	If device readied in the meantime, branch to..... >	PR75			
	An IPW$WFI call is issued to wait for a possible card reader interrupt.				IPW$WFI
	On card reader interrupt, indicate a new job stream in the card reader.				
PR75	If no STOP condition has been posted in the TCB, a branch is made to open the interface again..... >	PR02			
PR80	Using register 12 as a work register, the address of the task terminator is stored in the TCB, register 9 is loaded with the terminator base address, and a branch is made to terminate the task.		TCRC(IPW$DTC)	R12 R9	
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.				IPW$GAM
	If device readied in the meantime, branch to..... >	PR88			
	Wait for card reader interrupt.				IPW$WFI
	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			



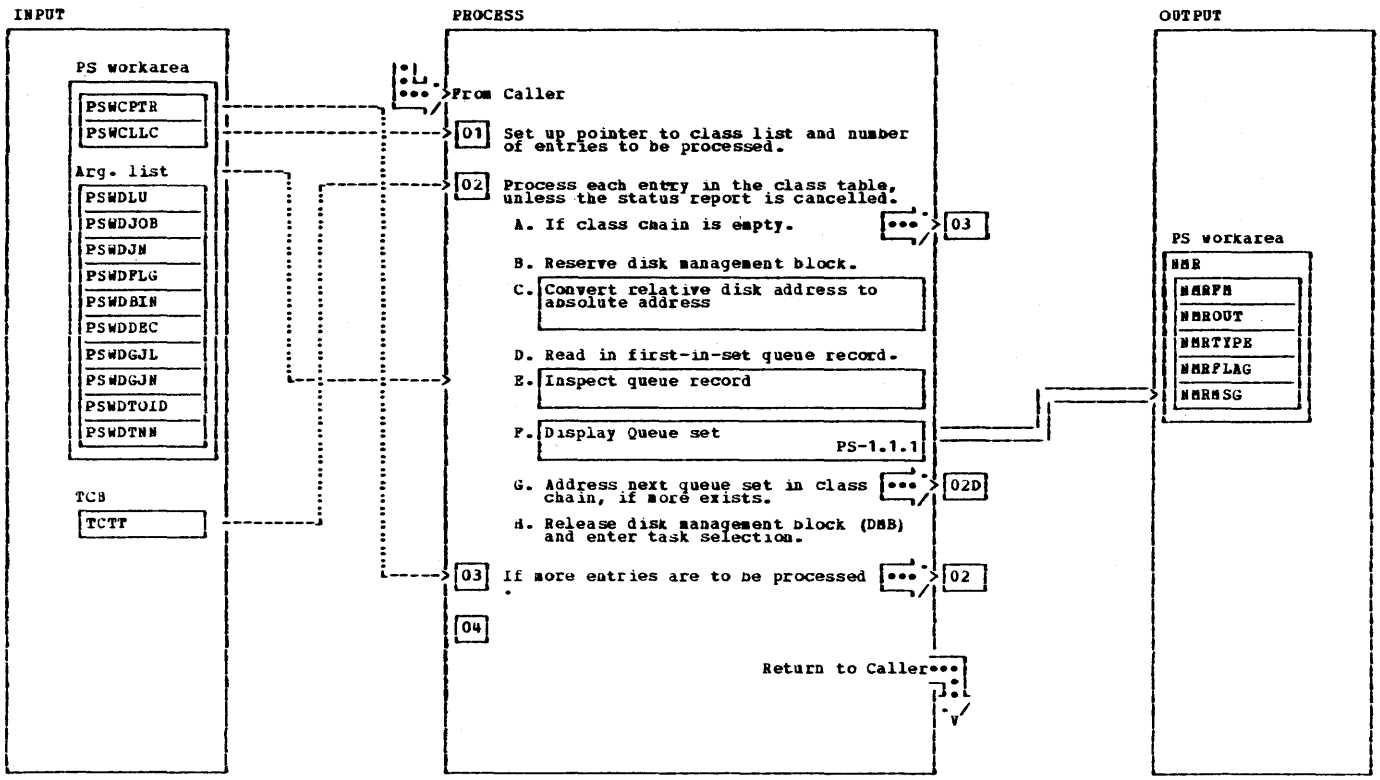

IPW$$PS - Print Status
INPUT
PROCESS
OUTPUT
TCB
TCTT
TCPL
TCPL
CPEA
PS workarea
PSWFLG1
Arg. list
PSWDID
PSWDLD
From PDISPLAY (in IPW$$CD) via task selection
01 Entry Processing
A. Reserve and initialize work area
B. If CTLSPOOL and queue display
- open I/F to log. reader
- pass initial record to logical reader
- build dataset header record
- pass record to logical reader
- update queue record
C. Set up CCB and CCW in workarea if status report should go to line printer
- advance to next page on printer
D. Obtain SYSLOG device type
E. Set up nodal message record (NMR)
F. If not Pdisplay PNET Command.
04
02 Process Pnet Display.
Process PNET Display. PS-1.2
03
05
04 Process Queue Display.
Process Queue Display. PS-1.1
05 If print status has not been cancelled.
07
06 Inform operator that print status report is incomplete.
07 Exit Processing
A. Advance to next page, if output of report was destined for local printer.
B. If output of report was spooled to disk as LST queue entry:
- indicate end of report
- close logical interface.
C. Unassign programmer logical unit, if applicable.
D. Post processing complete, if an ECB was supplied by requestor.
E. Release workarea and detach task.
08
Return to VSE/POWER task selection
PS workarea
Arg. list
CCB
CCW
NMR
NMRFM
NMROUT
NMRTYPE
NMRFLAG

IPW$$PS - Print Status

	NOTES	MODULE	LABEL	REF
1A	Workspace is acquired and initialized: - storage descriptor - the argument list, built by the command processor is copied into the workarea			$RSW
1B	If the print status report is supposed to be spooled to disk as LST queue entry. the interface to the logical reader is opened.			$OLI
1B	A skip to channel one is issued in order to advance to the next page. it also causes that the logical reader allocates a new queue entry for the LST entry being built.			$PLR
1B	The queue record, built by the logical reader, is set up with following values: - jobname - LST queue id - starting number of pages (1) - device type (dummy) Additionally the SPL (spool parameter list) is set up with information from the LST queue entry: - jobname and number			
1C	If the output of the print status report is destined for a printer, which has been already assigned to by the PDISPLAY command processor a CCB and write-CCW is set up in the work area. A skip to channel one is issued in order to advance to the next page on the printer.			

	NOTES	MODULE	LABEL	REF
1D				$ULP
6	If the status report has been cancelled by the operator by means of the PCANCEL command, message 1R46I STATUS REPORT CANCELED BY OPERATOR is iussed.			$GAH
7B	If the output is of the print status report is spooled to disk as LST queue entry, which is the case when a CTLSPOOL is issued by a cross partition user, the logical reader is informed to add the queue entry being built to the LST queue.			$PLR
7B	finally the logical interface is closed.			$CLI
7C	If a programmer logical unit was assigned to the printer destined for the output of the report, it is now unassigned, unless the logical unit is SYSLST.			$ULP
7E	The workarea is released and the storage is returned to the VSE/POWER storage pool.			$RLW
7E	The task is removed from the task selection list.			$DET

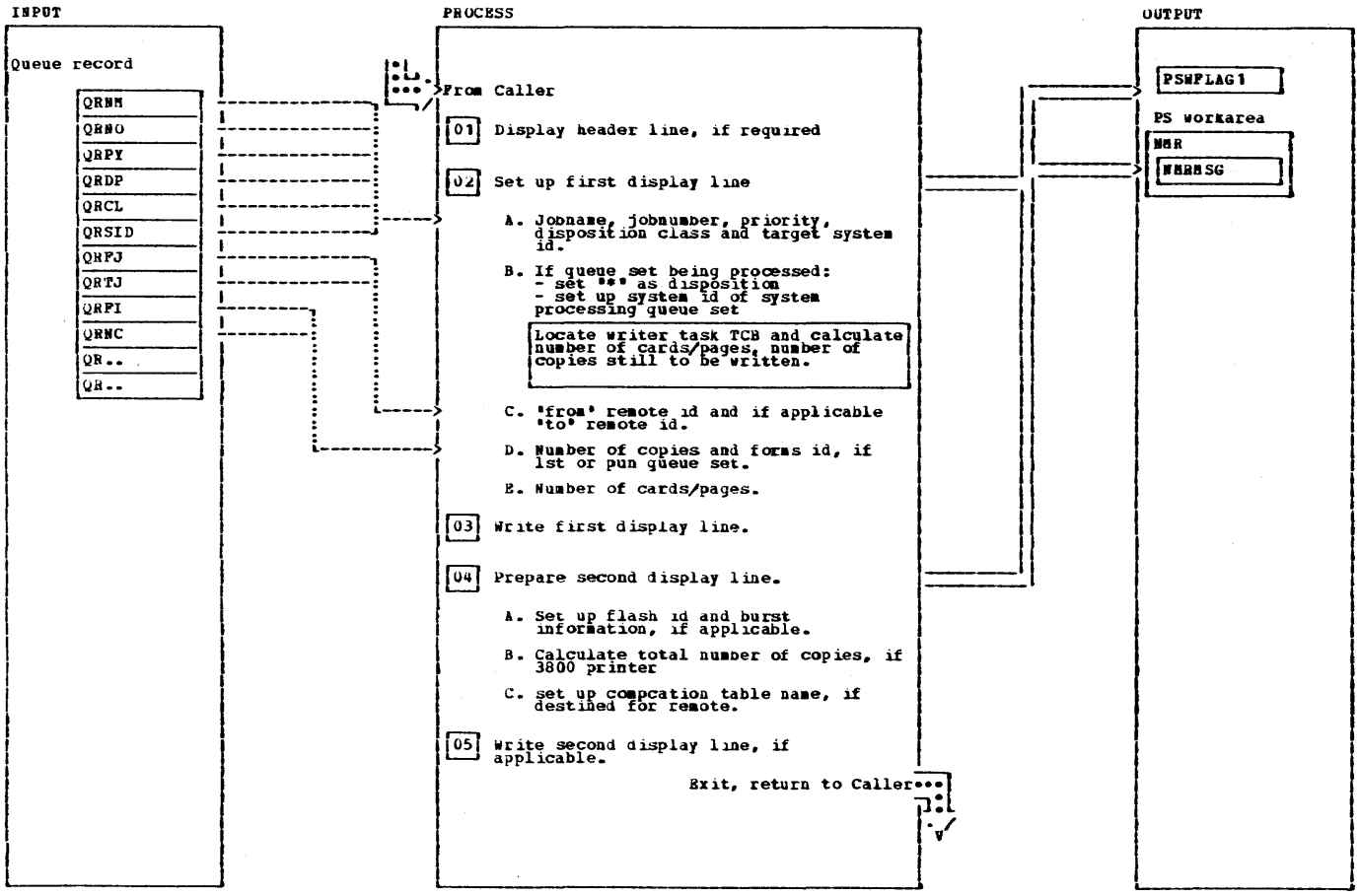
IPW$$PS - Process Queue display



	NOTES	MODULE	LABEL	REF
	The reader , list , punch and/or xmit queue is scanned and each queue set which is eligible is extracted and displayed.			
2B	The disk management block is reserved for the duration of the queue scanning.			$RSR
2D	The first in set queue record is read in the auxiliary queue record area located in the disk management block (DMB).			$RDQ
2E	The queue record just read in is examined if it is eligible to the displayed. The argument list has been prepared by the PDISPLAY command processor. Argument can be: - jobname and number - class - RJE user id - local or remote destination - target node name.			

	NOTES	MODULE	LABEL	REF
2G	Queue sets belonging to a class are chained via the 'QCQN' field. This field contains the absolute disk address of the next queue set in chain. For the last queue set, this field contains binary zeros.			
2H	The disk management block is released.			$RLR
2H	Task selection is forced to give other task with a higher priority than this one and waiting for the DMB, the chance to get the disk management block.			$WFD

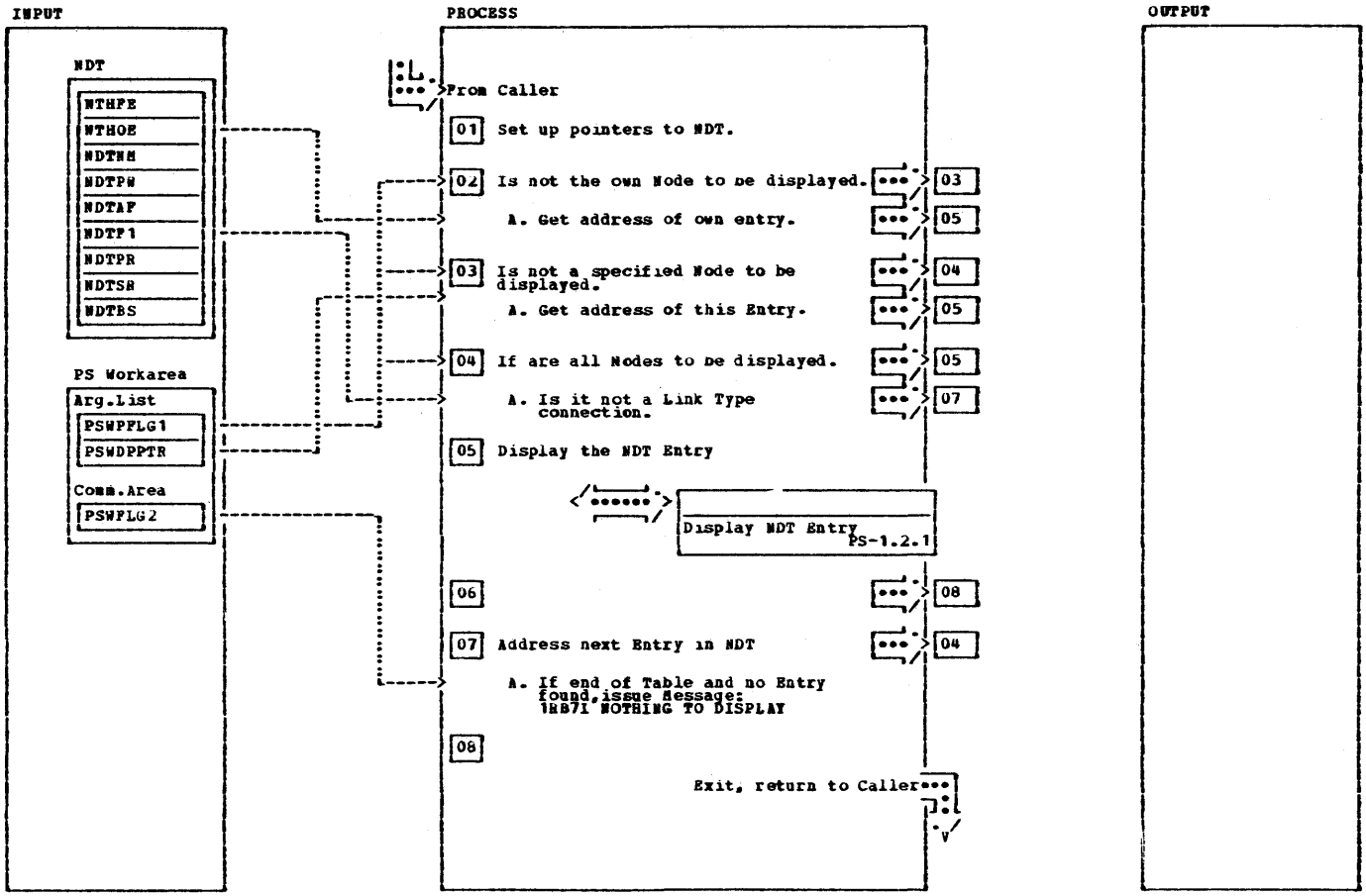
IPW$$PS - Display Queue set



NOTES	MODULE	LABEL	REF
Relevant information of the queue set being read in are displayed in the first and if applicable a second display line. In detail, that is: - jobname and number - priority, disposition and class - number of cards/pages - number of copies (LST|PUN only) - forms-id (LST|PUN only) - from / to remote id - target system id - compaction table name - flash id, burst info and copy groupings (3800 printer only .		PSDPLAY	
1 A header is printed on top of each queue display. The header is already set up in the display area.			
2B AN '*' is set as disposition to indicate that the queue set is just being processed.			
2B If the queue set is being processed by a physical writer task (LST=PUN), the TCB of that task is located, because it contains in the account workarea the most valid, still to do, number of copies as well as number of pages or cards, respectively.			

NOTES	MODULE	LABEL	REF
2D In the case a physical writer task has been stopped prior with the restart option, the remaining, still to do, number of copies is taken.			
2D If the queue set is a LST entry and the output is destined for a 3800 printer, the copy value is preceeded by an '*', to indicate that it represents a transmission count rather than the total number of copies. Note: the total number of copies is displayed in the 2nd display line.			
3 The first display line is written using the 'MSG' subroutine, additionally a flag is set which prevents that the header line is issued again.			
4A The flash as well as the burst id are only displayed if present.			
4C If the queue set is a LST entry and the output is destined for a remote terminal and a compaction table name was specified, the name is displayed in the 2nd display line.			

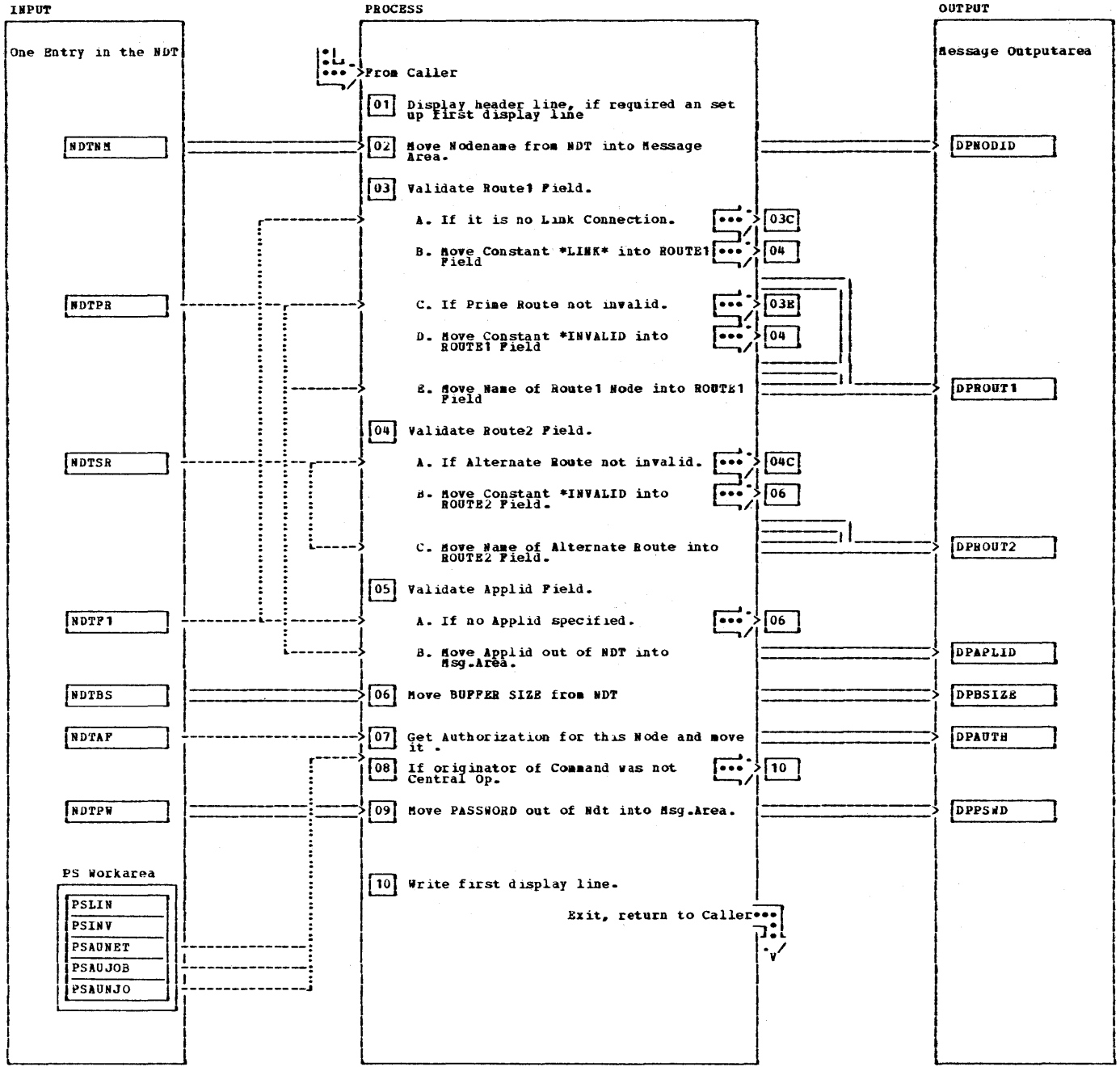
IPW$$PS - Process PNET Display



NOTES	MODULE	LABEL	REF
2 Flagbyte PSWPFLG1 contains: x'80' if the own Node has to be displayed.			
3 Flagbyte PSWPFLG1 contains: x'20' if specified Node has to be displayed.			
3A Address already stored in Argumentlist by CD.			

NOTES	MODULE	LABEL	REF
4 Flagbyte PSWPFLG1 contains: x'10' if all Nodes are to be displayed.			
4A Flagbyte NDTF1 contains: x'80' if the Node is a Link Connection.			
7A			$GAM

IPW$$PS - Display NDT Entry



NOTES	MODULE	LABEL	REF
Relevant information of the NDT entry(s) are displayed in the first and if applicable a second display Line. In detail, that is: - Name of the Node - Name of the ROUTE1 Node - Name of the ROUTE2 Node - Authorization - Buffer Size - Application Id - Password (if originator was Local Op.)			
1 A header is printed on top of the display. The header is already set up in the display area.			

NOTES	MODULE	LABEL	REF
3A Flag NDTF1 contains x'80' if it is a *LINK*.			
3C NDTPR contains x'00' if invalid.			
4 NDTSR contains x'00' if invalid.			
7 Flagbyte NDTAF indicates which authorization is generated for this Node.It can be : x'F0' System Authorization. x'50' Net Authorization. x'10' Job Authorization. x'00' Nojob Authorization.			
10 The Msg-Subroutine will be used to to write the Display Line.			

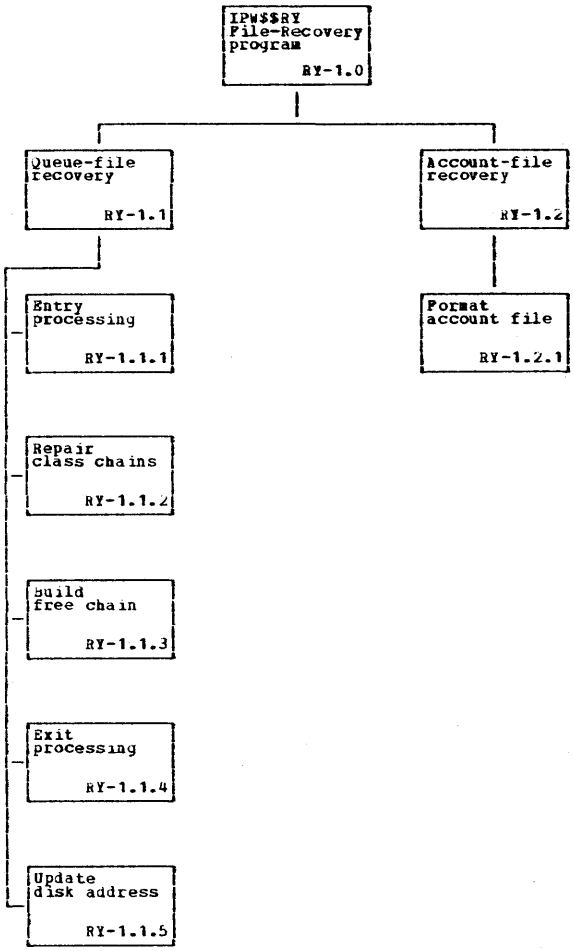
IPW$$RQ - Reserve Queue Record	
Label	Routine
RQ00	Function Entry
RQ04	Test for Record Space
RQ20	Reserve Queue Record
RQ40	Function Exit

Services Used	
Service	Macro
Task Management	IPW$WFC
Resource Management	IPw$RLR IPW$RSR
Storage Management	IPW$RSW
Message Service	IPW$GAM
Disk / Tape Service	IPw$CTT IPW$RDQ IPw$WTQ IPW$WTT
Timer Service	IPW$RDC

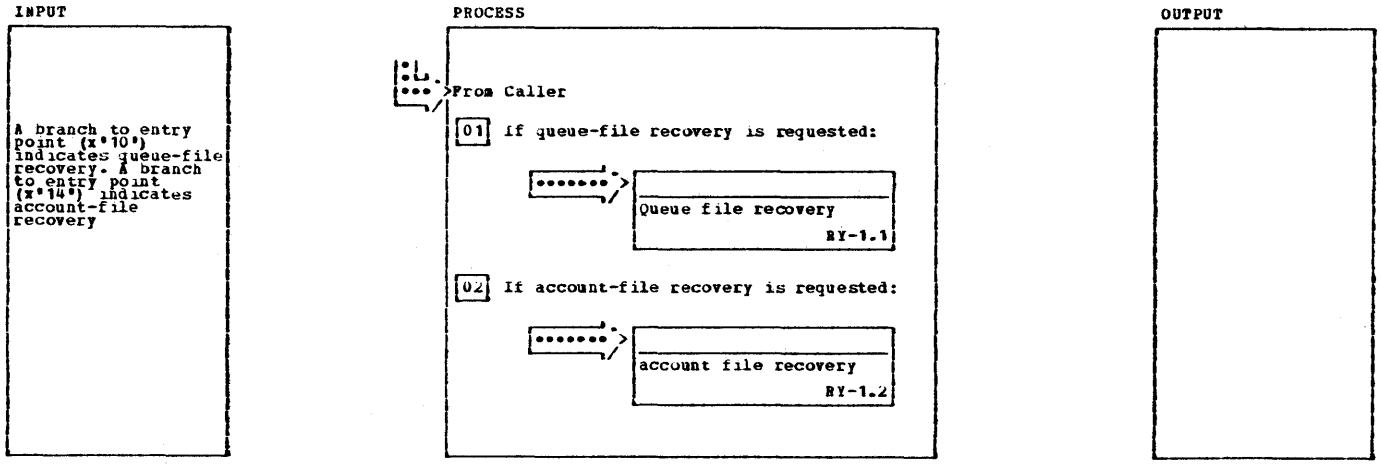
Called by	
Module	Description
IPW$$LR	Logical Reader
IPW$$NR	Network Receiver
IPW$$OF	Offload Queues
IPW$$XW	Execution Writer

Labels	Chart RQ: IPW$$RQ - Reserve Queue Record		Modified Data Fields	Reg. Usage	Calls
RQSD	The first 16 bytes constitute the section descriptor:				
	'RQCS release'				
RQ00	Function Entry				
	Save caller registers 14 through 9 inclusive.		IPW$DSV		
	Set function track byte to 'R'.				
	Reserve queue record space (if not available).				IPW$RSW
	Save addresses.		TCQA(IPW$DTC) TCQV(IPW$DTC)		
RQ02	If no tape spooling, branch to.....>	RQ04			
	Set first-in-set switch		QRFS(IPW$DQR)		
	Write initial tapemark.				IPw$CTT
	Write the record to tape (queue record area).				IPW$WTT
	Exit.....>	RQ44			
RQ04	Set addressability disk management block (DMB).			R6	
	Lock DMB.				IPw$RSR
	Examine Free List Pointer in Master Record				
RQ05	If no queue record available:				
	Issue message 1Q38I NO DASD SPACE AVAILABLE				IPW$GAM
RQ10	Unpost ECB in DMB.		QCEB(IPW$DQC)		
	If save account task insert cancel code 'c' and call terminator routine		TCTT(IPW$DTC)		
RQ12	Unlock DMB.				IPW$RLR
	Wait for posting of ECB in DMB.				IPw$WFC
	Lock DMB.				IPw$RSR
	Examine free list pointer. If zero, branch to retry.....>	RQ10			

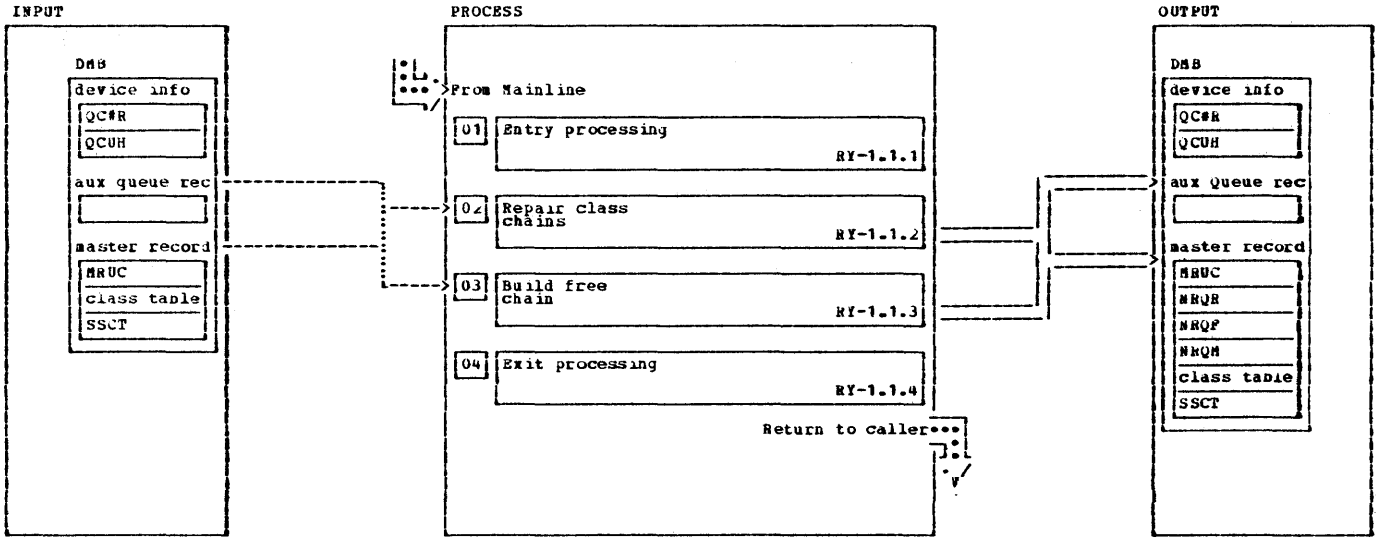
Labels	Chart RQ: IPW$$RQ - Reserve Queue Record		Modified Data Fields	Reg. Usage	Calls
RQ20	Reserve Queue Record				
	Read first free queue record from set		QCQW(IPW$DQC)		IPW$RDQ
	If queue-record traps are active and the record is not flagged as a free record, force a program check.....>	RQSD			
	Update first free queue record to next record.		MRQF(IPW$DQC)		
	Update queue record field:				
	set first-in-set switch		QRFS(IPW$DQR)		
	set next-in-set pointer zero		QRNS(IPW$DQR)		
	set forward chain pointer zero		QRQP(IPW$DQR)		
	set previous chain pointer to queue record itself		QRQN(IPW$DQR)		
	Set own system-id		QRSY(IPW$DQR)		
	Write record back to file				IPW$WTQ
	Get current time		QRST(IPW$DQR)		IPW$RDC
	Set as current start and stop time		QRET(IPW$DQR)		
	Update record counters in DMB (statistical information).				
	• Decrement number of queue records available		NRQF(IPW$DQC)		
	• Maximum number of records in use.		NRQM(IPW$DQC)		
	• Number of tracks (1 for CKD or unit of transfer for FBM)		QRNT(IPW$DQR)		
RQ40	Function Exit				
	If it is not save-account task do not unlock DMB.....>	RQ44			
	Unlock DMB.				IPW$RLR
RQ44	Set function track indicator to 0 (open for output).		TCTF(IPW$DTC)		
	Restore registers and return to caller.	R14		R14-R9	



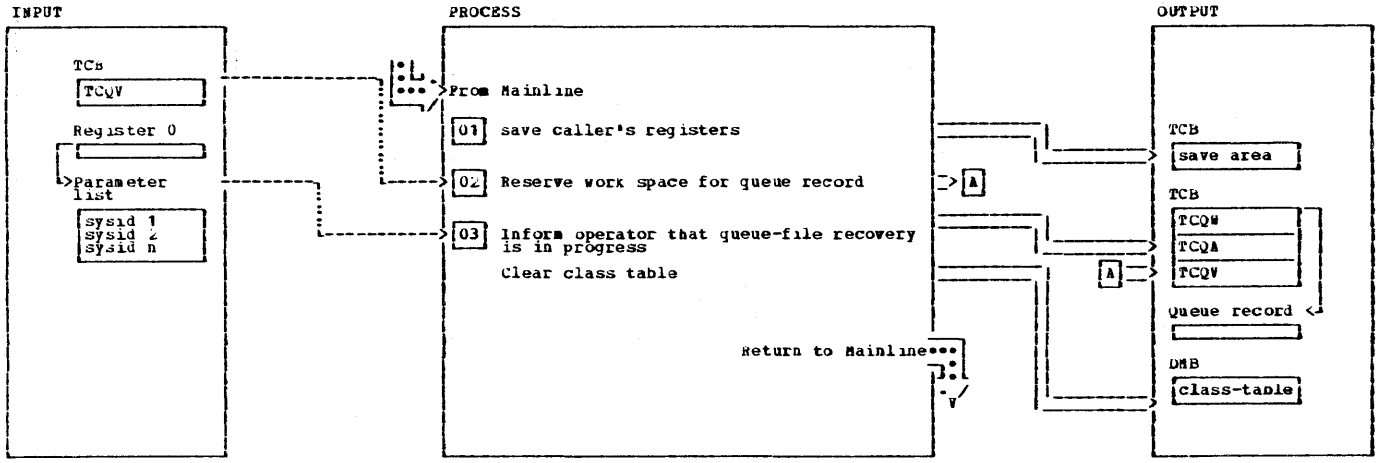
File-Recovery Processing



Queue-file-recovery processing



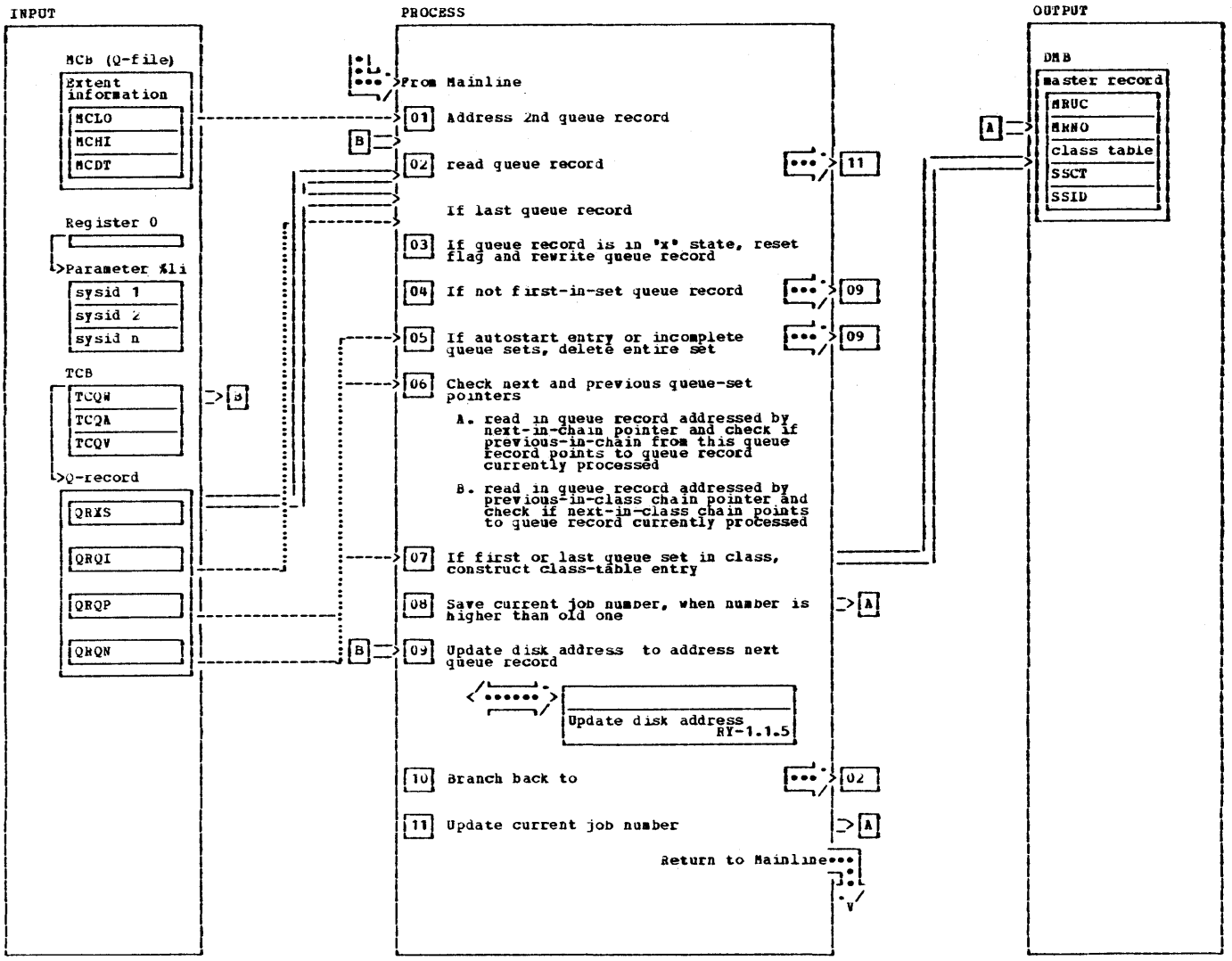
Recovery program entry processing



NOTES	MODULE	LABEL	REF
1 The caller's registers are saved in our own save area, addressed by register 13. Register 0 contains the address of a parameter list containing the one-byte SYSIDs of the systems which abnormally terminated the last time. This has been recognized either by the timer task or at initialization time. The function-track byte in the task control block is set to 'Y' to indicate queue-file recovery in progress.		RY000	$SAV

NOTES	MODULE	LABEL	REF
2 If no queue space is currently held by the task, work space is reserved and the real / virtual addresses are saved in the TCB			$RSW
3 The operator is informed that queue-file recovery is being done. Message IQB7I QUEUE FILE RECOVERY IN PROGRESS In a shared-spooling environment, the message is extended by the SYSIDs of the system for which recovery is being performed			$GAM

Repair class-chain

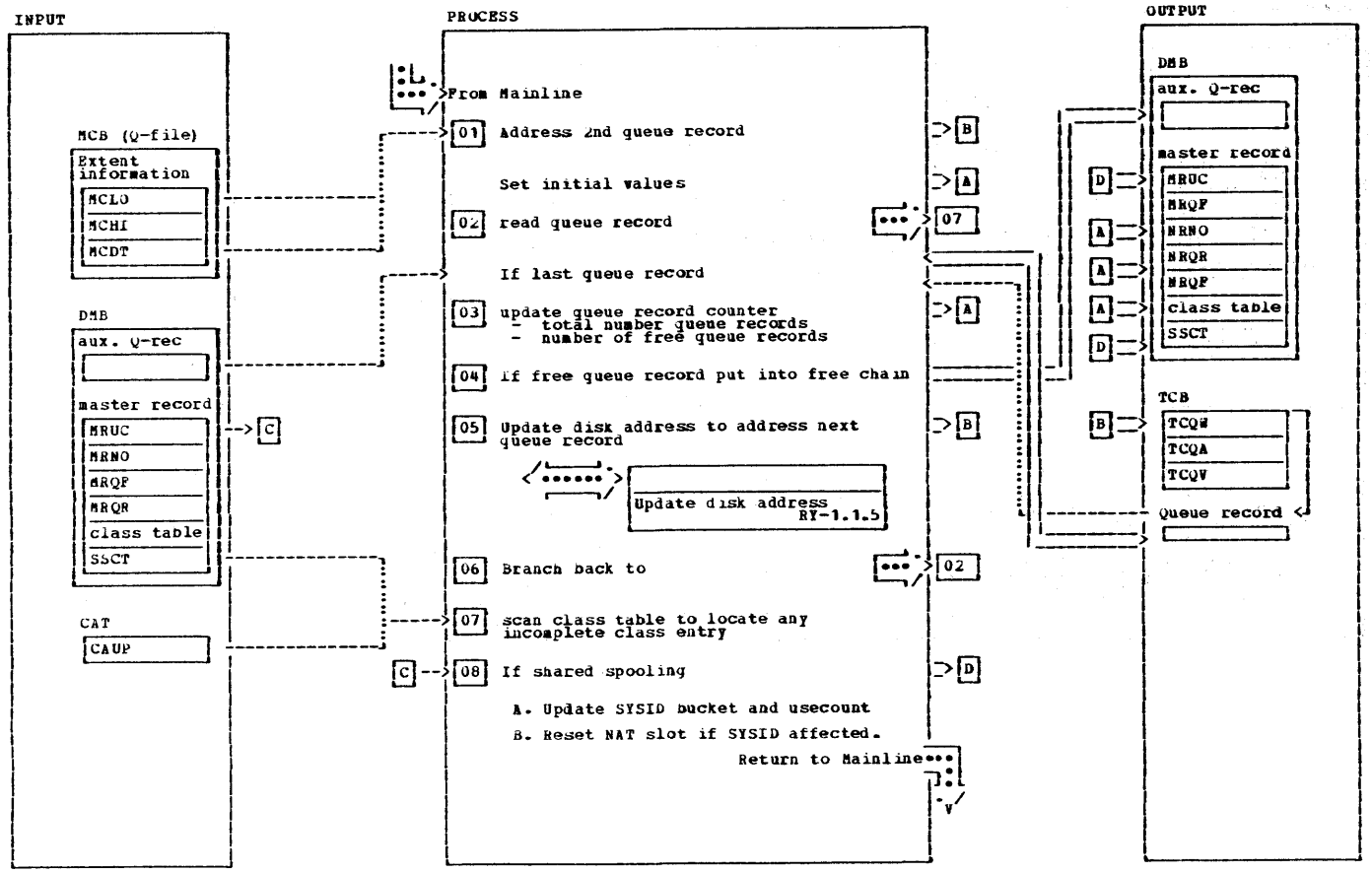


Repair class-chain

NOTES	MODULE	LABEL	REF
1 The second queue record in the file is addressed. This is done by using the lower limits of the extent and specifing Rec=2 for C-K-D or setting the relative block number to one for FBA The seek address has following format: MBBCCHHR for C-K-D or MOBBRRRR for FBA The first queue record on the queue file is reserved for internal purposes			
2 The queue record is read in the area addressed by 'TCQV' queue record identifier 'D'		RY050	$RDQ
3 Depending on the mode (either shared or non-shared) all queue records which are marked as being in execution state (if non-shared), or only the queue records which are being processed by system(s) abnormally terminated last time as indicated in the parameterlist passed by the caller, are reset.		RY060	
3 The execution flag is reset and the queue record is re-written			$WTQ

NOTES	MODULE	LABEL	REF
4 Ignore queue records which are not first-in-set queue records.			
5 An incomplete queue set is recognized when the previous queue-set pointer addresses the first-in-set queue record itself. The first-in-set queue record and all queue records that are chained to it are flagged as free queue records. The queue records are then written back to the queue file, so that they can be later put in the free chain.		RY100	$RDQ $WTQ
6 The next and previous queue set pointers (forward and backward) are checked to insure that the class chain is not broken. If so, the chain is repaired if possible. If not possible (next or previous queue-set pointer addresses a free queue record or a non-first-in-set queue record), message 1Q24I QUEUE FILE CHAIN ERROR is issued and VSE/POWER is terminated		RY130	$RDQ $GAM
7 The absolute disk address is converted to a relative queue-record number and put into the class table. The 'live' flag, indicating that at least one entry is in the the class-chain, is set			

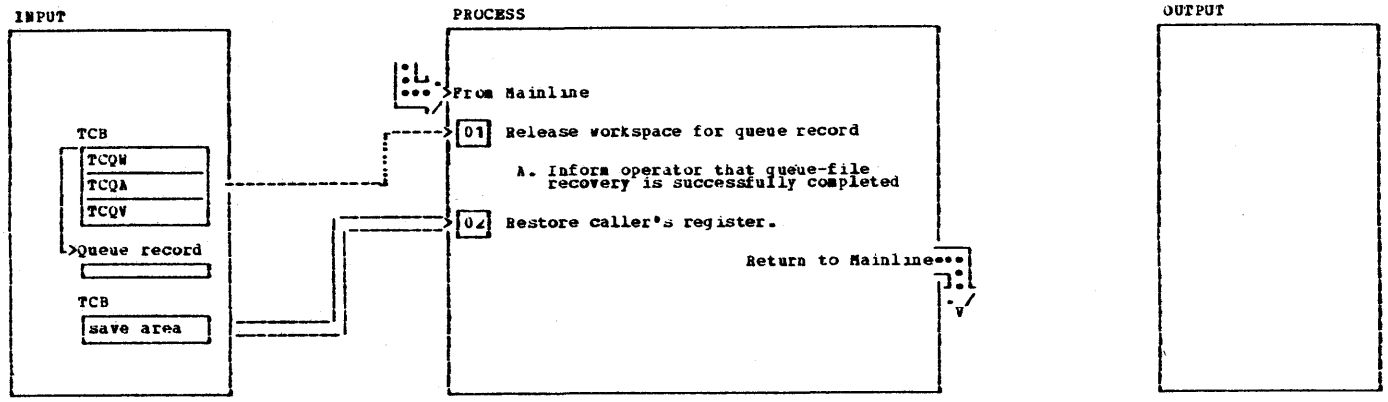
Build free chain



NOTES	MODULE	LABEL	REF
1 The second queue record in the file is addressed. This is done by using the lower limits of the extent and specifing Rec=2 for C-K-D or setting the relative block number to one for FBA		RY300	
The seek address has following format: MBBCCHHR for C-K-D or MOBBRRRR for FBA			
The first queue record on the queue file is reserved for internal purposes			
The free-queue-record disk address is set to zero and the number of queue records as well as the number of free queue records are set to zero as initial value			
2 The queue record, addressed by seek bucket 'TCQW' is read in.		RY310	
If the just-read queue record is the last one (queue id = 'D') and there is still a free queue record in the auxiliary-queue-record area in the DMB, the end of free chain indicator is set and the queue record is written back as last free queue record.			$WTQ
Queue record identifier 'D'			$RDQ
3 The total number of queue records is increased by 1			
4 When the queue record is marked as a free one, the free-queue-record count is increased by one and the record is added as last entry in the free-queue-record chain.			$WTQ

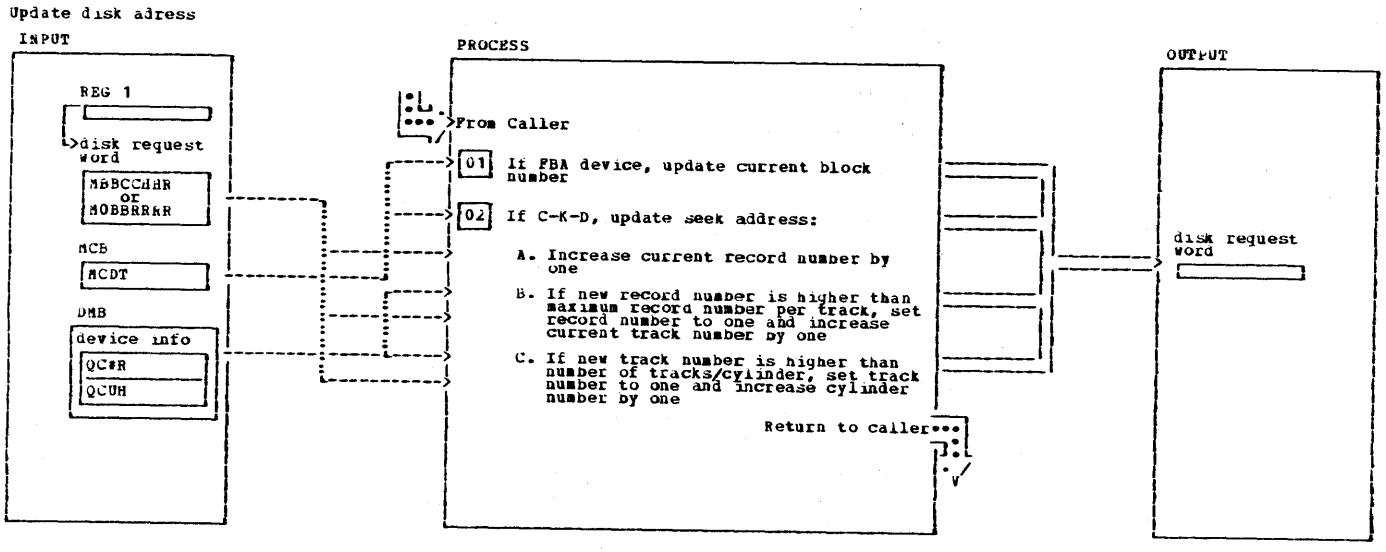
NOTES	MODULE	LABEL	REF
This is done by chaining the previous free queue record, which is saved in the auxiliary-queue-record area in the DMB, to the new one			
Then the previous free queue record is written back. Finally the new free queue record is saved in the auxiliary-queue-record area.			$WTQ
7 The class table is now scanned to locate any incomplete class chain (this is the case when a pointer exists to the first queue set, but no last queue-set pointer or vice versa). If this condition occurs and the queue file error trap option is set by user, message 1Q241 QUEUE FILE CHAIN ERROR is issued and VSE/POWER is terminated		RY460	GAM
Otherwise, option was not set, all queue records belonging to the incomplete class chain are removed, thereby deleting the class. However the queue records are not available anymore			
8 If shared spooling, the usecount is decremented by one and the SYS-ID, if specified is removed from the connected SYS's table. The Node Attach Table (NAT) in the DMB is searched to reset all entries related to the SYSID for which recovery is to be made.			

Recovery program exit processing



	NOTES	MODULE	LABEL	REF
1	The queue space, if present, is released and returned to the storage pool. The function track byte in the task control block is reset to . . to indicate successful completion of queue-file recovery.		RY900	$RLW

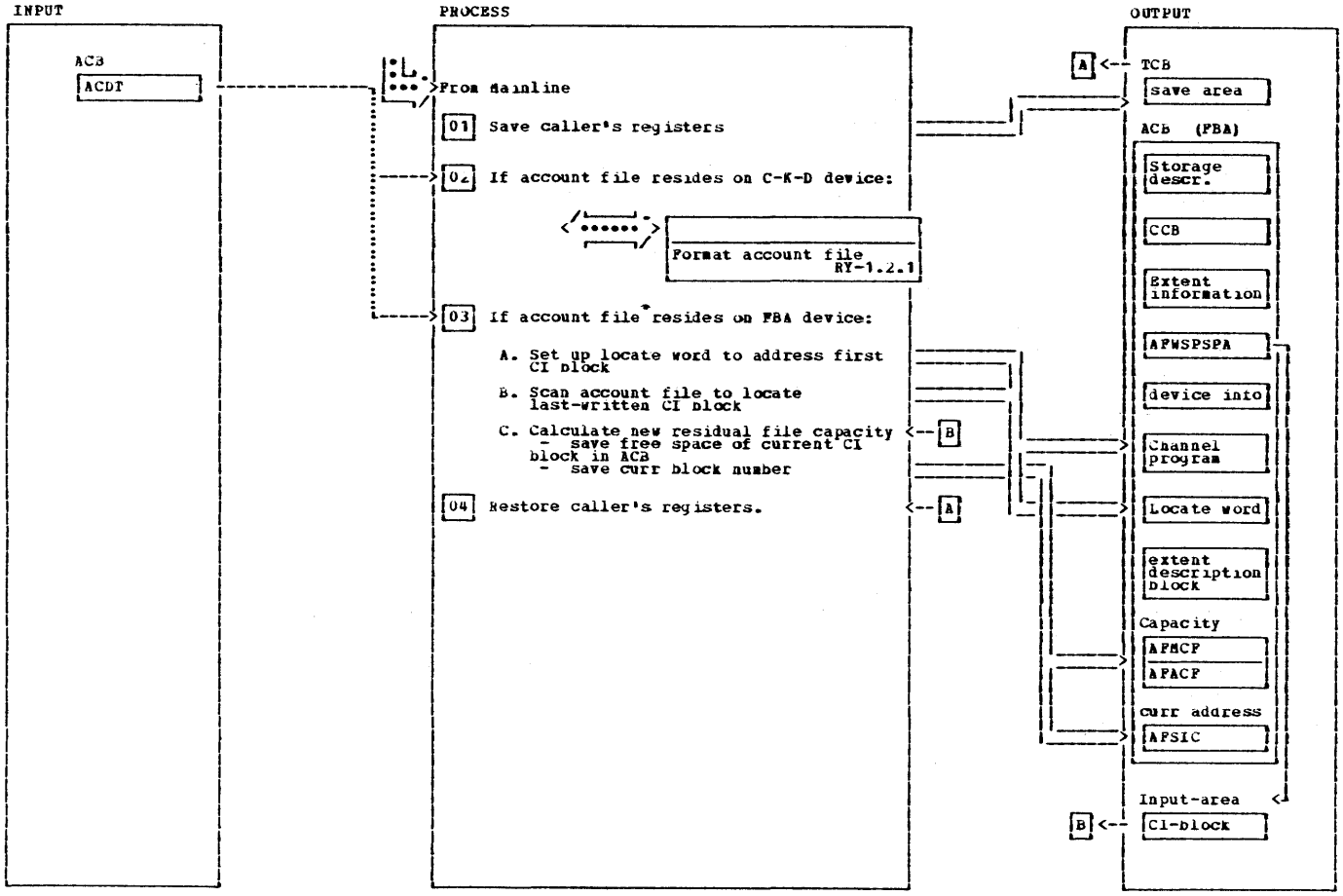
	NOTES	MODULE	LABEL	REF
1A	The operator is informed that queue-file recovery is completed. Message IQB8I QUEUE FILE RECOVERY COMPLETED is issued.			$GAM
2	The caller's register are restored from the save area, addressed by register 13.			$RET



NOTES	MODULE	LABEL	REF
This subroutine updates the disk address, pointed to by register 1, to address the next queue record. The seek address has following format: MBBCCHHR for C-K-D MOOBBBRRR for FBA		UPDT	

	NOTES	MODULE	LABEL	REF
1	When the queue file resides on a FBA device, the current block number, as stored in the disk request word, is increased by one and the result is stored back in the disk request word.			

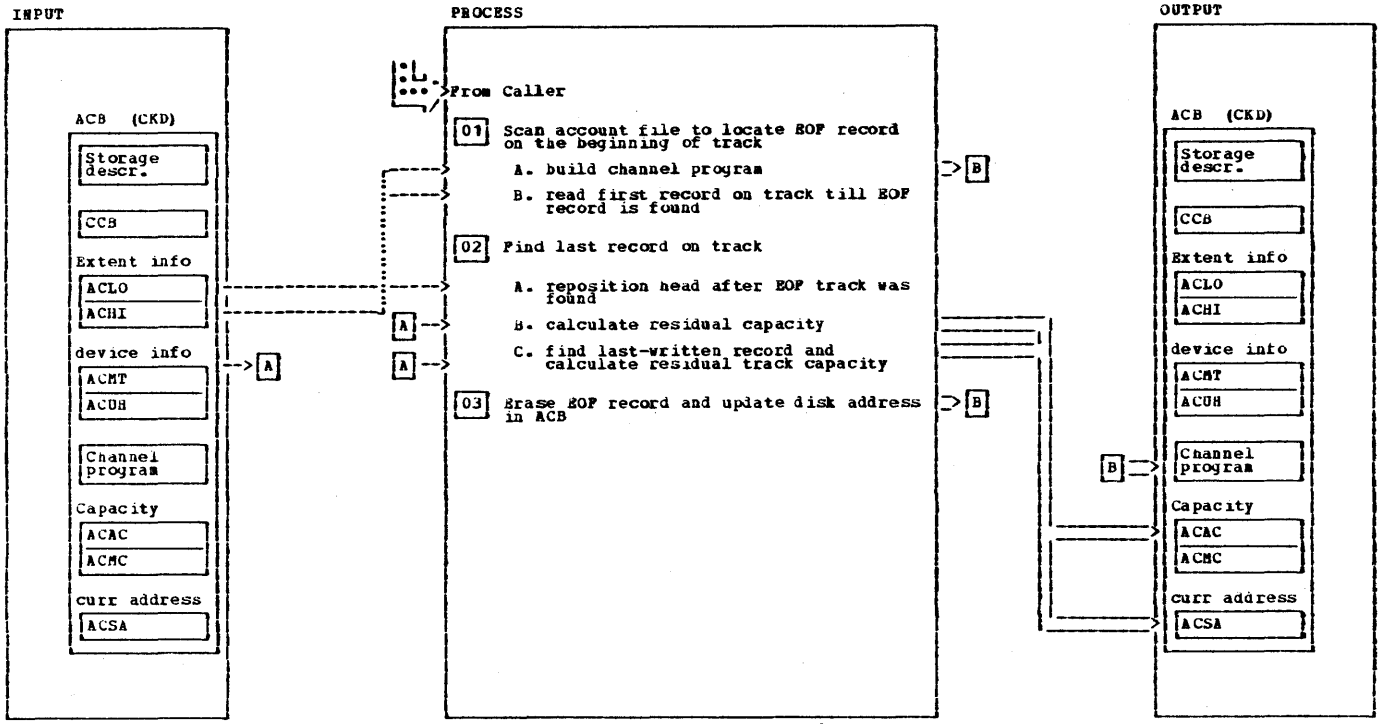
Account-file-recovery processing



NOTES	MODULE	LABEL	REF
When account-file recovery is invoked the account file is scanned to locate the last-written account record if C-K-D, or the last-written CI if FBA, respectively. The residual file and track/CI capacity are recalculated and stored in the ACB. Additionally, the disk address is saved in the ACB.			
1 The caller's registers are saved in a save area, addressed by register 13.		AF000	$SAV
3 The EOF-CI block is searched. This is done from the beginning of the extent (lower limit). If found, the relative blocknumber is used to address the last written and used CI block.			
3A The channel program in the ACB is prepared to read in the first CI block. The locate word is set up with following values: - inhibit write - indicate read only - CI block number = 0		AF500	

NOTES	MODULE	LABEL	REF
3B The account file is read sequentially and each CI block read in is examined if it is the EOF CI	EXCP	AF520	
3B If not, the next CI block is addressed in the locate word and the CI block is read in	WAIT		
3C The residual capacity is calculated and saved in the ACB. Additionally the free space in the current CI block as well as the CI block number are saved in the ACB.		AF600	
4 The caller's register are restored from the save area addressed by register 13		AF950	$RET

Format account file (C-K-D)



	NOTES	MODULE	LABEL	REF
1A	The standard channel program, set up at VSE/POWER initialization time and placed in the ACB, is modified to read the count field of the first account record of each track. The original channel program is saved previously. - seek CCW - TIC or Set sector - read data - read count - read sector (if RPS is supported)		AP100	
1A	The seek address of the first record on the track is initialized with the lower-limit value and rec=0 is set.			
1B	The first record on each track is read in and examined if it is the EOF record. If not, the seek bucket is updated, unless end-of-extent is reached, to address the first record on the next track. Then the record is read in.	EXCP	AP120	$WFC

	NOTES	MODULE	LABEL	REF
2A	When the EOF track was found and the file is not empty (that is when the first track contains EOF record), the seek address is repositioned (one track backwards) to address the previous track			
2B	The residual file capacity is calculated by subtracting the number of tracks in use from the total number of tracks. (Note: the free space on the last track in use is not considered.) Now the channel program is modified in order to read the count field and the sector value, if applicable, of the first record on the track.			
2C	Each record on the last track in use is read without data transfer and examined to determine if it is the EOF-record. When the EOF record is hit, the residual track capacity is calculated.	EXCP	AP400	$WFC
2C 3	The original channel program is restored	GETVCE EXCP		$WFC

IPW$$SA - VSE/POWER Save Account	
Label	Routine
SA00	Function Entry
SALR	Spool Account File to Punch Queue
SADL	Delete Account File
SATP	Save Account File to Tape
SADA	Save Account File to Disk
SACL	Function Exit
SAGT	Read Account Record
SAPL	Pass Account Record to Punch Queue

Services Used	
Service	Macro
Task Management	IPW$DET IPW$WFC
Resource Management	IPW$RLR IPW$RSR
Storage Management	IPW$RLV IPW$RLW IPW$RSV IPW$RSW
Message Service	IPW$GAM

Interfaces used	
Module	Macro
IPW$$LW	IPW$PLR
IPW$$NU	IPW$CLI
IPW$$NU	IPW$OLI

Functions used	
Module	Macro
IPW$$AS	IPW$IAS
IPW$$GA	IPW$CAF
IPW$$GA	IPW$GAR
IPW$$GA	IPW$OAF
IPW$$LU	IPW$ULP

Called by	
Module	Description
IPW$$CJ	Command Processor (PACCOUNT)

Labels	Chart SA: IPW$$SA - Save Account		Modified Data Fields	Reg. Usage	Calls
SASD	The first 16 bytes constitute the section description:				
	'SACS release'				
	On entry, the following register contents are relevant:				
	1: Device type and LUB unit output file			R1	
	The following registers are used by IPW$$SA:				
	8: Account file CCB address			R8	
	9: SACS base register			R9	
	10: Address of VSE/POWER 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: Base register			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$RSW
	If the Save Device specified is DEL, a branch is made to erase the account file..... >	SADL			
	The account file is opened in GET mode.				IPW$CAF
	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 SA: IPW$$SA - Account Save		Modified Data Fields	Reg. Usage	Calls
SA04	If device is not a tape, exit ..... >	R14			
	Otherwise, the LUB index in register 5 is stored in a dummy CCB, and branch to unassgn tape ..... >	SC04	PHLU	R8	
SA08	If the account file has to be written to tape, a branch is made ..... >	SATP			
	If it has to be saved on disk, a branch is made ..... >	SADA			
	Otherwise, the account file is spooled via the logical reader to punch queue.				
	This routine spools the account file via the logical reader to the punch queue.				
SALR	Register 5 is loaded with the address of the IPW$$SA output area.			R5	
	The logical reader interface is opened.				IPW$OL1
	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 SA: 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.		ACRD		
	• 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.		ACRD		
	• Record length is set to zero and made negative to signal end of account record.			R6	
	Branch to .....>	LR20			
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	

Labels	Chart SA: IPW$$SA - Account Save		Modified Data Fields	Reg. Usage	Calls
	Residual data length in register 6 is decremented by the length of the data just moved.			R6	
LR20	Branch if not first spool record... >	LR24		R3	
	Otherwise, provide logical reader with dummy read command (2nd hopper, MFCU 5425).		TCCC(IPW$DTC)		
	Set register 1 to one as minimum record length.			R1	
	A link is made to pass this dummy record..... >	SAP4			
	(The address of the output card image is loaded in register 0 by the subroutine SAPL.)				
	On return, the dummy read command is cleared.		TCCC(IPW$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.		QRNM(IPW$DQR)		
	• Record identifier is set to Punch.		QRQI(IPW$DQR)		
	• Class indicator is set to Punch.		QRCL(IPW$DQR)		
	• Priority to set to 1.		PRPY(IPW$DQR)		
	• Disposition is set to HOLD.		QRCL(IPW$DQR)		
	• The default number of separator cards is moved to the queue record.		QRSP(IPW$DQR)		
	Handle data set header record .....>	LRDSHR			
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			

Labels	Chart SA: IPW$$SA - Save Account		Modified Data Fields	Reg. Usage	Calls
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		
	Card count in queue record is updated with one for this EOJ card, using register 3 as work register.		QRLC(IPW$DQR)	R3	
	A link is made to pass the EOJ record to the logical reader..... >	SAPL		R2	
	Register 1 is set to zero to simulate EOF condition of the physical reader.			R1	
	A link is made to pass the EOF condition to the logical reader.... >	SAP1		R2	
	The logical reader interface is closed.				IPW$CLI
	Normal exit is taken..... >	SC08			
SADL	This routine is entered when the account file is to be erased.				
	The DMB and ACB are reserved.				IPW$RSR
	A IPW$CAF ERASE is issued to erase the account file.				IPW$CAF
	The DMB and ACB are released.				IPW$RLR
	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 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,		

Labels	Chart SA: 1PW$$SA - Save Account		Modified Data Fields	Reg. Usage	Calls
	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			
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.		TCSADY(1PW$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		

Labels	Chart SA: IPW$$SA - Save Account		Modified Data Fields	Reg. Usage	Calls
TP12	The LUB index (programmer logical unit number) is moved to the CCB.		PHLU		
	The tape DTF is opened, invoking VSE/POWER Asynchronous Service..... >	SAOPEN			
	If unrecoverable I/O error is posted in the TCB (by Terminator routine IPW$$TR), or open was not successful, 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			
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		PH1S		
	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.				

Labels	Chart SA: IPW$$SA - Save Account		Modified Data Fields	Reg. Usage	Calls
	The account record is written to tape, with suppression of incorrect length.			R8	
	A IPW$WFC VSE/POWER wait macro is issued to wait for I/O completion.			R1	IPW$WFC
	If no unit exception (EOV) on tape is detected, a branch is made to process the next account record..... >	TP21			
	On EOV:				
	• FEOV switch is set on.		PH1S		
	• EOF switch is set off.		PH1S		
	• A link is made to SAOPEN routine to invoke VSE/POWER Asynchronous Service to process EOV..... >	SAOPEN		R0,R1	
	• The alternate tape is opened. If open not successful .....>	TP14			
	• The block count is reset to zero.				
	• The next account record is processed..... >	TP21		R5	
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.			R5	
	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		

Labels	Chart SA: IPW$$SA - Save Account		Modified Data Fields	Reg. Usage	Calls
	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.		SAWC+4		
	The DASD DTFPH is now opened: • Work space is reserved for the SRB and it is initialized. • Asynchronous services is called to perform the OPEN. • The SRB work space is released.				IPW$IAS IPw$RSW IPW$IAS IPW$RLW
	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		

Labels	Chart SA: IPW$$SA - Save Account		Modified Data Fields	Reg. Usage	Calls
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	
	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.)				

Labels	Chart SA: IPW$$SA - Save Account		Modified Data Fields	Reg. Usage	Calls
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		
DA13	Seek and search arguments are copied from count field.		SASA		
	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.			R1	
			SACF		
	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.			R1	
			SACF		
	The record number is set to zero.		SACF		

Labels	Chart SA: IPW$$SA - Save Account		Modified Data Fields	Reg. Usage	Calls
	Seek and search arguments are copied from the count field.		SASA		
DA20	Using register 1 as a work register, record number in count field is incremented by one.		SACF		
	Next sector value is set for the set sector CCW. On EOF account file... >	DA24	SASE		
DA21	The account record is written.				
	A IPW$WFC macro is issued to wait for I/O completion.				IPW$WFC
	On completion, branch to get the next account record..... >	DA08			
DA24	This is the DASD normal (EOF) exit routine.				
	Set key and data lengths to zero and update seek address .....>	DA13	SACF		
	The write data CCW is unchained.		SAWC		
	An EXCP is issued to write the EOF record.				
	An IPW$WFC macro is issued to wait for I/O completion.				IPW$WFC
	On completion, the address of message 1Q79I is loaded in register 4.			R4	
	A branch is made to exit..... >	SC08			
DA28	This routine is a DASD error exit routine.				
	The address of message 1Q81I is loaded in register 4.			R4	
	The user file name is moved into message 1Q81I.		FLNM		
	Branch to error exit..... >	SC12			
SACL	This routine is the common close and exit routine. The output file is closed; its device is unassigned; if no errors have occurred, the account file is erased; a message is logged stating the action taken by IPW$$SA; the IPW$$SA task is detached.				

Labels	Chart SA: IPW$$SA - Save Account		Modified Data Fields	Reg. Usage	Calls
	The (tape) output file is closed:				IPW$FCH
	• Workspace is reserved and initialized for the SRB.				IPW$RSW
	• Asynchronous service is called to perform the close.				IPW$IAS
	• The SRB workspace is released.				IPW$RLW
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.				
	If no other density was specified ..>	SC06			
	The default mode in the PUB is set to standard.		PUBOPTN	R1	
SC06	Register 1 is loaded with the VSE/POWER PIB address.			R1	
	Register 3 is loaded with the CCB address of the output file and its device is unassigned.			R3	IPW$ULP
	If the open tape file procedure was unsuccessful, branch to..... >	SC12			
	If the OPEN account file procedure was unsuccessful, a branch is made to message logging routine (1Q78I)..... >	SC16			
SC08	Otherwise, the account file is closed and erased.				IPW$CAF
	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
SC16	The address of the message is stored in register 1 and the DTFPH pointer is saved in register 4 and the message is issued.			R1,R4	IPW$GAM IPW$WTO
	The DTFPH work space is released.				IPW$RLW
	The TCB account track indicator is set inactive (X'40') and the task is detached.		TCAT(IPW$DTC)		IPW$DET
SAGT	This subroutine reads an account record from the account file.				
	An account record is read.				IPW$GAR
	IPW$GAR will return the account record length in register 0, and the account record address in register 1.			R0 R1	

Labels	Chart SA: IPW$$SA - Save Account	Modified Data Fields	Reg. Usage	Calls
	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	
SAPL	This routine passes a card image record via the logical reader to the punch queue.			
	The card sequence number is increased by one and stored in the record.	SQNO		
	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
	Control is passed back to the calling routine.		R2	

Labels	Chart SA: IPW$$SA - Save Account		Modified Data Fields	Reg. Usage	Calls
	Handle Data Set Header Record (DSHR):				
LRDSHR	Save registers 0-3		LRSAVE	R0,R1	
	Reserve storage in the length of the DSHR. If not successful ....>	LRDSHRE		R1	IPW$RSV
	Initialize the DSHR with the following values:				
	• Total record length.		NDHLEN		
	• Length of general section.		NDHGLEN		
	• General section ID and modifier.		NDHGTYPE,NDHGMOD		
	• Fixed length record format (X'82').		NDHGRCFM		
	• Record length of 80.		NDHGLREC		
	Set the DSHR and control record flags		TCG2,TCCC		
	Pass the record to logical reader.				IPW$PLR
LRDSHRE	Reset the DSHR and control record flags and reload registers 0-3.		TCG2,TCCC		
	Return to caller.....>	R2			
SAOPEN	Save return register 2.		TCSART		
	Reserve space for SRB.				IPW$RSW
	Reload return register 2 and setup open request code.		SRBREQ(IPW$DSR)		
	If end-of-volume processing required>	SAOPEN1			
	Setup parameter list with DTF address and OPENR phasename.				
SAOPENR	Call asynchronous services.				IPW$IAS
	Release SRB workspace.				IPW$RLW
	Reload return register and return to caller .....>	R2			
SAOPEN1	Setup parameter list with DTF address and EOV phasename, branch to .....>	SAOPENR			

IPW$$SC - VSE/POWER Scan Reader JECL Statement	
Label	Routine
SC00	Determine Positional/Keyword Parameter Format
SC20	Keyword Routine
SC50	Positional Format Routine
CL00	CLASS Parameter Routine
JN00	JNM Parameter Routine
DI00	DISP Parameter Routine
PY00	PRI Parameter Routine
US00	USER Parameter Routine
DE00	DEV Parameter Routine
FI00	FID Parameter Routine
NO00	NOD Parameter Routine
VS00	VSC Parameter Routine
VE00	VER Parameter Routine
FD00	FEED Parameter Routine
PW00	PWD Parameter Routine
ID00	SYSID Parameter Routine
XE00	XDEST Parameter Routine
LS00	LDEST Parameter Routine
PU00	PDEST Parameter Routine
NT00	NTFY Parameter Routine

Called by	
Module	Description
IPW$$LR	Logical Reader

Labels	Chart SC: IPW$$SC - Scan Reader JECL Statement		Modified Data Fields	Reg. Usage	Calls
SASD	The first 16 bytes constitute the section descriptor:				
	'SCCS release'				
	The following register contents are relevant at entry:				
	0: address of end of statement to be checked			R0	
	1: address of parameter to be checked			R1	
	5: address of queue record		IPW$DQR	R5	
	10: address of VSE/POWER permanent area		IPW$DPA	R10	
	11: address of TCB		IPW$DTC	R11	
	13: address of task save area		IPW$DSV	R13	
	15: base address IPW$$SC			R15	
	Define Format:				
SC00	Caller's registers are saved.				IPW$SAV
	Parameter registers 0 and 1 are saved in registers 6 and 7, respectively.			R6 R7	
SC02	Register 4, to be used in subsequent error indication, is set to 0.			R4	
	Register 3 is set up to contain the machine length of the field to be scanned for the current parameter, being the field starting at the parameter address passed and ending at the statement end.			R3	
	With a translate and test instruction, the delimiter of the parameter is found.			R1,R2	
	If no delimiter found, branch to check form switch.....>	SC12			
	The value, stored in register 2 by the TRT instruction, is now used as a branch index to the appropriate routine:				
SC05	4: '=' delimiter, branch to process keyword form.....>	SC20			
	8: ',' delimiter, branch to process positional form.....>	SC50			
	12: ' ' delimiter, branch to process last parameter.....>	SC10			
	16: ')' delimiter, error branch ...>	SC90			
SC10	The last parameter switch is turned ON.		LWPI(IPW$DTC)		
	A branch is then made to process positional parameter.....>	SC50			

Labels	Chart SC: 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	LWP1(IPW$DTC)		
	Keyword Form Handling Routine:				
SC20	Temporarily saving the delimiter address in register 3, the form switch, as previously set, is examined using a translate and test instruction, which loads register 2 with the appropriate branch index.			R3 R1 R2	
	Using register 2 as branch index register, a branch is now made to the appropriate routine:				
SC22	4: 'keyword', branch to continue..>	SC30			
	8: 'positional' branch to process error situation.....>	SC26			
	12: blank (not set), branch to process first keyword parameter.....>	SC24			
	16: Logic error, branch .....>	SC5D			
SC24	Forms switch is set to 'keyword'.		LWFS(IPW$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 SC: 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.		LWB1(IPW$DTC)		
	Register 7 is set to point to the start of the parameter value.			R7	

Labels	Chart SC: IPW$$SC - Scan Reader JECL Statement		Modified Data Fields	Reg. Usage	Calls
SC36	Register 1 is set to point to 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	
	If the real length is zero, branch to set last parameter switch.....>	SC38			
	Otherwise, scan for the next delimiter (comma or blank)			R1,R2	
	If not found before end of statement, branch to indicate last parameter...>	SC38			
	The scan 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 .....>	SC36			
	8: ',' delimiter, branch to .....>	SC45			
	12: ' ' delimiter, branch to .....>	SC40			
	16: ')' delimiter, branch to .....>	SC45			
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			
SC45	If 1st byte of parm. not '(' .....>	SC90			
	Load pointer to '(' and branch .....>	SC36			
SC46	If 1st byte of parm. '(', branch ...>	SC48			
SC47	If delim. is cc72, branch .....>	SC40			
	If blank after delim. ',', branch ..>	SC40			
SC48	If no previous delim. occurred .....>	SC36			
SC49	If previous delim. was ',', error ..>	SC90			
	Check for cc72 and blank, branch....>	SC47			
	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	

Labels	Chart SC: IPW$$SC - Scan Reader JECL Statement		Modified Data Fields	Reg. Usage	Calls
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)		
	Parameter Syntax Check Routines:				
SC70	Parameter address in register 1 is saved. Branch index is loaded in register 2 from the TCB.			R1 R2	
SC72	The branch index will now control branching to the appropriate parameter check routine:				
SC100	Branch to CLASS parameter routine.....>	CL00			
SC101	Branch to JNM parameter routine.....>	JN00			
SC102	Branch to DISP parameter routine....>	DI00			
SC103	Branch to PRI parameter routine.....>	PY00			
SC104	Branch to USER parameter routine....>	US00			
SC105	Branch to DEV parameter routine.....>	DE00			
SC106	Branch to FID parameter routine.....>	FI00			
SC107	Branch to NOD parameter routine.....>	NO00			
SC108	Branch to VSC parameter routine.....>	VS00			
SC109	Branch to VER parameter routine.....>	VE00			
SC110	Branch to FEED parameter routine....>	FD00			
SC111	Branch to XDEST parameter routine...>	XD00			
SC112	Branch to LDEST parameter routine...>	LD00			
SC113	Branch to PDEST parameter routine...>	PD00			
SC114	Branch to NTFY parameter routine....>	NT00			
SC116	Branch to PWD parameter routine.....>	PW00			
SC117	Branch to SYSID parameter routine...>	ID00			

Labels	Chart SC: IPW$$SC - Scan Reader JECL Statement		Modified Data Fields	Reg. Usage	Calls
	Class Parameter Syntax Check Routine:				
CL00	If not JOB statement, branch	CL02			
	If statement has keyword form	CL05			
	Otherwise, set the last parameter		LWPI(IPW$DTC)		
	switch ON, and branch	CL05			
CL02	If not CTL statement, branch	SC90			
CL05	If CLASS parameter not already				
	specified in current statement	CL10			
	If already specified, and current				
	statement is CTL statement, branch to				
	flag invalid parameter.....	SC90			
	If JOB statement, reset CLASS value		QRCL(IPW$DQR)		
	in queue record to default.				
	Register 7 is loaded with address of			R7	
	start of keyword, and branch	SC90			
CL10	CLASS parameter identifier switch is		LWPI(IPW$DTC)		
	set ON. The parameter length is set				
	in register 3.				
	If parameter length zero, branch ...	SC92			
	Otherwise, if more than one, branch				
	to check two characters.....	CL30			
	A one-character CLASS parameter				
	should be any of the following				
	characters: A-Z, or 0-9.				
	If CLASS parameter indicates				
	autostart class, branch to change				
	default class indicator in queue				
	record accordingly.....	CL20			
	If CLASS parameter is not alphameric,				
	or if numeric higher than 9, or lower				
	than 'A', branch to flag invalid ...	SC90			
	If current statement is not a CTL				
	statement, branch to change default				
	class indicator in queue record to				
	value specified.....	CL20			
	If current statement is a CTL				
	statement, change default class value				
	in TCB.		TCCT(IPW$DTC)		
	Branch to return.....	SC92			
CL20	Default class value in current queue		QRCL(IPW$DQR)		
	record is set according to class				
	value specified. 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.				

Labels	Chart SC: IPw$$SC - Scan Reader JECL Statement		Modified Data Fields	Reg. Usage	Calls
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			
	If current statement is in positional form, branch to flag invalid parameter.....>	SC90			
CL40	If parameter is not a supported partition, branch to flag invalid parameter.....>	SC90			
	Else, change default class value in queue record to '0' (for BG) or '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			

Labels	Chart SC: IPW$$SC - Scan Reader JECL Statement		Modified Data Fields	Reg. Usage	Calls
JN10	The JNM parameter switch is set ON.		LWPI(IPW$DTC)		
	The parameter length is calculated in register 3.			R3	
	If zero (omitted), branch to return.....>	SC92			
	Using register 2 as a work register, the parameter length is compared with the maximum allowable length of 8 bytes. If longer than 8 bytes and not a DOS/VSE job card, branch to flag invalid parameter.....>	SC90		R2	
	Register 3 is decremented by one to contain the parameter length in machine format, and the parameter is scanned for non-alphamerics.			R3	
	If non-alphameric characters present, branch to flag invalid parameter....>	SC90			
	Otherwise, the job name field in the queue record is blanked out.		QRNM(IPW$DQR)		
	The JNM parameter is copied into the job name field.		QRNM(IPW$DQR)		
	Branch to return.....>	SC92			
	PWD Parameter Syntax Check Routine:				
PW00	If current statement not JOB statement, branch to flag invalid parameter.....>	SC90			
	If PWD parameter not already specified, branch to check PWD parameter.>	PW10			
	Else set default password of zero and branch to invalid parameter routine.>	SC90	QRPW(IPW$DQR)	R7,R3	
PW10	Set password parameter indicator. The parameter length is calculated in register 3. If zero (omitted) branch to return.....>	SC92			
	If longer than 8 bytes, flag as invalid.....>	SC90			
	Register 3 is decremented by 1 to contain the parameter length in machine format and the parameter is scanned for non-alphamerics. If non-alphamerics present, branch to flag invalid parameter.....>	SC90			
	Otherwise the password field in the queue record is blanked and the new password moved in. Branch to return.....>	SC92	QRPW(IPW$DQR)		

Labels	Chart SC: IPW$$SC - Scan Reader JECL Statement		Modified Data Fields	Reg. Usage	Calls
	SYSID Parameter Syntax Check Routine:				
ID00	If current statement not JOB statement, branch to flag invalid parameter.....>	SC90			
	If SYSID not already specified, branch to check SYSID parameter.....>	ID10			
	Otherwise indicate no target processor.		QRSID(IPW$DQR)		
	R7 is loaded with the start of the keyword and a branch is made to flag invalid parameter.....>	SC90		R7,R3	
ID10	Set SYSID parameter indicator.				
	The parameter length is calculated in R3.			R3	
	If zero (omitted) branch to return..>	SC92			
	If longer than 1 byte, flag as invalid.....>	SC90			
	If no target-id wanted set SYSID in queue record to zero and return.....>	SC92	QRSID(IPW$DQR)		
ID15	Check that the SYSID is between 1 and 9. If not, flag as error.....>	SC90			
	Otherwise update SYSID in queue record and return.....>	SC92	QRSID(IPW$DQR)		
	DISP Parameter Syntax Check Routine:				
DI00	If current statement is not a JOB statement, branch to flag invalid parameter.....>	SC90			
	Branch index in TCB is set for PRI, which is the next positional JOB parameter.		LWBI(IPW$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$LQR)		
	The address of the start of keyword DISP is loaded in register 7, and a branch is made to flag invalid parameter.....>	SC90		R7	

Labels	Chart SC: IPW$$SC - Scan Reader JECL Statement		Modified Data Fields	Reg. Usage	Calls
DI10	The DISP parameter identifier switch is set ON.		LWPI(IPW$DTC)		
	The parameter length is calculated in register 3.			R3	
	If zero (parameter omitted), branch to return.....>	SC92			
	If greater than one, branch to flag invalid parameter.....>	SC90			
	Otherwise, if the DISP parameter is D (for dispatchable), H (for hold), K (for keep), or L (for leave), branch to continue.....>	DI20			
	Otherwise, branch to flag invalid parameter.....>	SC90			
DI20	The DISP parameter value is moved to the queue record.		QRDP(IPW$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$LTC)		
	If PRI parameter not already specified, branch to continue.....>	PY10			
	Otherwise, using register 3 to address the DMB, the default priority value is copied from DMB to queue record.		QRPY(IPW$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 SC: 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$DTC)		
US20	Parameter pointer register 1 is saved in register 9.			R1 R9	
	If USER parameter not already specified, branch to bypass setting keyword pointer register 7.....>	US40			
US30	Register 7 is decremented to point to the keyword start address.			R7	
US40	User information field in the queue record is blanked out again.		QRUI(IPW$LQR)		
	USER parameter identifier switch is set ON, and a branch is made to flag invalid parameter.....>	SC90	LWPI(IPW$DTC)		

Labels	Chart SC: 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$DQR)		
	Branch to return.....>	SC92			
	DEV Parameter Syntax Check Routine:				
DE00	If the current statement is not a RDR statement, branch to flag invalid parameter.....>	SC90			
	Otherwise, the branch index in the TCB is set for FID, being the next positional RDR parameter.		LWBI(IPW$DTC)		
	Indicate data file mode processing in the TCB.		LWER(IPW$DTC)		
	If the DEV parameter has not already been specified, branch to continue >	DE10			
	If the queue record has not been reserved, branch to.....>	DE05			
	Otherwise, SYSIN mode processing is indicated in the queue record or the current job.		QRER(IPW$DQR)		

Labels	Chart SC: IPW$$SC - Scan Reader JECL Statement		Modified Data Fields	Reg. Usage	Calls
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$DPW)		
	Translate the EBCDIC value into packed hexadecimal form (X'0cuu'), and get it into register 2.		PEDW(IPW$LPW)	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			

Labels	Chart SC: IPW$$SC - Scan Reader JECL Statement		Modified Data Fields	Reg. Usage	Calls
	FID Parameter Syntax Check Routine:				
FI00	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.		LWB1(IPW$DTC)		
	If parameter length is zero (omitted).....>	FI15			
	If the first character of the FID parameter is not an apostrophe, branch to flag invalid parameter....>	SC90			
	Otherwise, register 4 is loaded with the address of the file ID in the physical work space, register 2 is set to 8 (maximum string length), and a link is made to pass these register values to the string scan routine...>	SS00		R2,R4	
	The string scan routine scans the quoted FID information string, and copies a valid string to the physical work space. It passes control back to the caller if the string is valid. If it is invalid, a return is made to an address 4 bytes passed the normal point.				
	If the FID string is valid, branch to continue.....>	FI10			
	If it is invalid, branch to diagnose the error.....>	FI20			
FI10	The parameter pointer register 1 is saved in register 9.			R9	
	If the FID parameter has already been specified, branch to flag invalid parameter.....>	FI30			
FI15	Otherwise, the FID parameter identifier switch is set on, and a branch is made to return.....>	SC92	LWP1(IPW$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	

Labels	Chart SC: IPW$$SC - Scan Reader JECL Statement		Modified Data Fields	Reg. Usage	Calls
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	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			
	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	

Labels	Chart SC: IPW$$SC - Scan Reader JECL Statement		Modified Data Fields	Reg. Usage	Calls
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	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$DTC)		
VS10	If the VSC parameter has not already been specified, branch to continue..>	VS20			
	Otherwise, set the VSC information field in the physical work space to the default value.		PESC(IPW$DPW)		
	Register 7 is decremented to point to the keyword start address, and a branch is made to flag invalid parameter.....>	SC90		R7	

Labels	Chart SC: IPW$$SC - Scan Reader JECL Statement		Modified Data Fields	Reg. Usage	Calls
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$DPW)		
	VER Parameter Syntax Check Routine:				
VE00	If the current statement is not a RDR statement, branch to flag invalid parameter.....>	SC90			
	If the VER parameter has not already been specified, branch to continue >	VE10			
	Otherwise, set the VER information field in the physical work space to the default value.		PEVE(IPW$DPW)		
	Register 7 is decremented to point to the keyword start address, and a branch is made to flag invalid parameter.....>	SC90		R7	

Labels	Chart SC: IPW$$SC - Scan Reader JECL Statement		Modified Data Fields	Reg. Usage	Calls
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)		
	FEED Parameter Syntax Check Routine:				
FD00	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	

Labels	Chart SC: IPW$$SC - Scan Reader JECL Statement		Modified Data Fields	Reg. Usage	Calls
FD10	The FEED parameter identifier switch is set on.		LWPI(IPU$DTC)		
	In register 3 is the length of the parameter, if zero, branch	SC92		R3	
FD20	If the length is greater than two characters, branch to.....	FD30			
	If the parameter is not 'NO', ...	SC90			
	Clear option FEED in PWS, branch ...	SC92	PEOP(IPW$DPW)		
FD30	If the length of the parameter is greater than 3, branch to.....	SC90			
	If the parameter is not 'YES', branch	SC90			
	Set the option FEED in PWS and branch	SC90	PEOP(IPW$DPW)		
	XDEST Parameter Syntax Check:				
XD00	If not a JOB statement or if the XDEST parameter has already been specified, branch	SC90			
	Check the parameter and if correct store it in the job header	PAND	NJHGXEQN NJHGXEQU		
	Move XDEST parameters to queue record		QRTN,QRTU		
	Set XDEST parameter identify switch.	SC92	LWPI3(IPW$DTC)		
	LDEST Parameter Syntax Check:				
LD00	If not a JOB statement, or if the LDEST parameter has already been specified, branch.....	SC90			
	Check the parameter and if correct store it in the job header	PAND	NJHGPRTN NJHGPRTR		
	The LDEST parm. switch is set ON Branch.....	SC92	LWPI3		
	PDEST Parameter Syntax Check:				
PD00	If not a JOB statement, or if the PDEST parameter has already been specified, branch.....	SC90			
	Check the parameter and if correct store it in the job header	PAND	NJHGPUNN NJHGPUNR		
	The PDEST parm. switch is set ON Branch.....	SC92	LWPI3		
	NTFY Parameter Syntax Check				

Labels	Chart SC: IPW$$SC - Scan Reader JECL Statement		Modified Data Fields	Reg. Usage	Calls
NT00	If not a JOB statement, or if the NTFY parameter has already been specified, branch.....>	SC90			
	Save NJHGORGR in NJHGUSID Check the parameter and if correct store it in the job header .....>	PAND	NJHGORGN NJHGUSID		
	If NTFY=YES specified, set NJHGUSID to the origin remote value (NJHGORGU)		NJHGORGR NJHGUSID		
	The NTFY parm. switch is set ON NJHGORGR is restored, then branch...>	SC92	LWPI3		
	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			

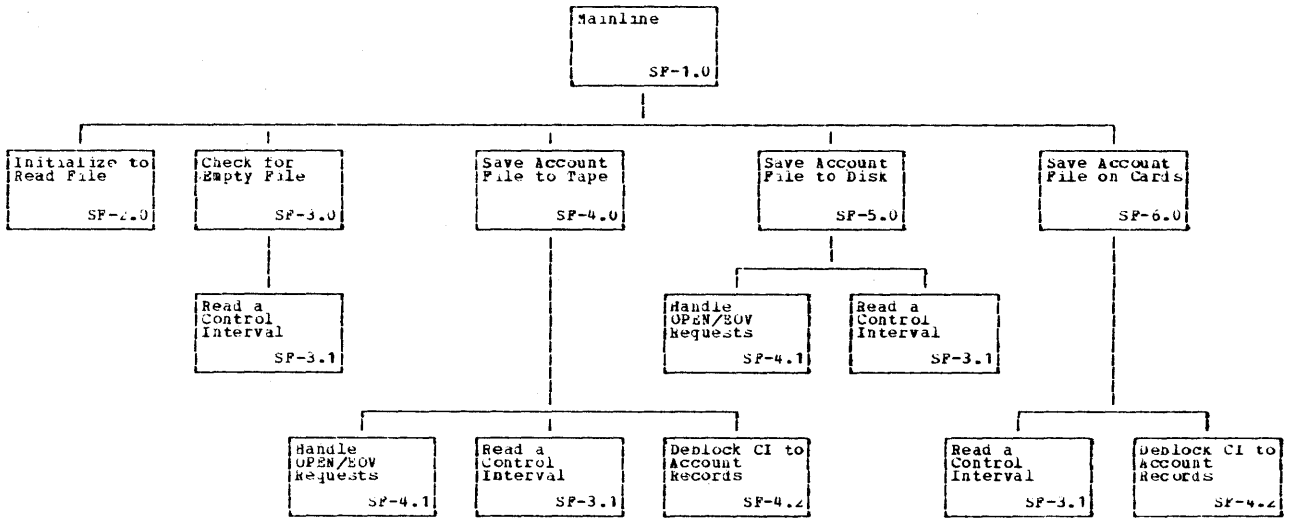
Labels	Chart SC: IPW$$SC - Scan Reader JECL Statement		Modified Data Fields	Reg. Usage	Calls
SS20	If string length greater than maximum allowed, branch to flush string.....>	SS10			
	Otherwise, move current character to queue record, and branch to get next character.....>	SS10	QRUI(IPW$DQR)		
SS30	Register 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	

Labels	Chart SC: IPW$$SC - Scan Reader JECL Statement		Modified Data Fields	Reg. Usage	Calls
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			
	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.		LWPL(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)			

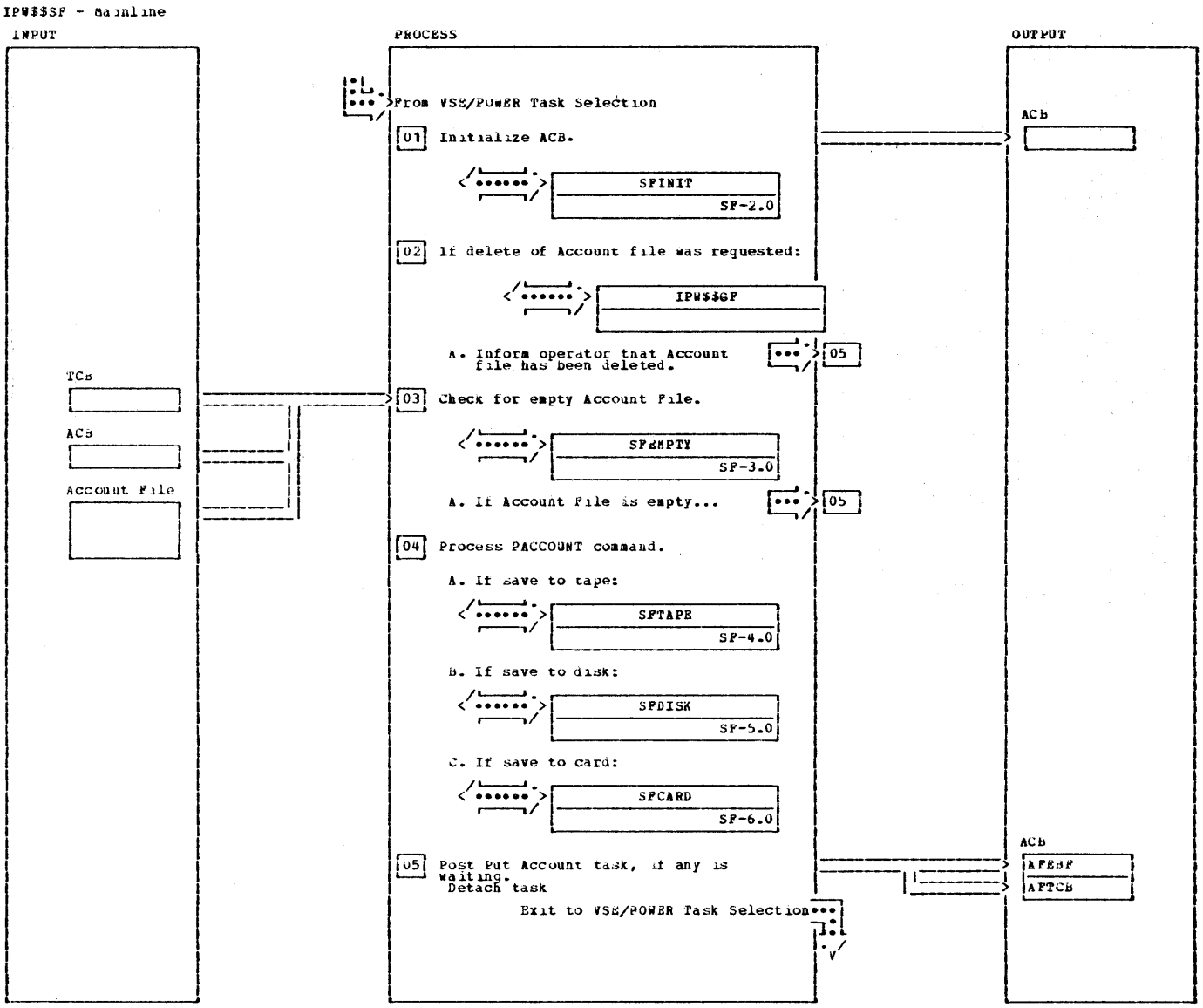
Labels	Chart SC: IPW$$SC - Scan Reader JECL Statement	Modified Data Fields	Reg. Usage	Calls
	Parsing Subroutine for Parameters in the format =(value,value)			
PAND	If 1st char. of parm. begins with '(', branch.....>	PANA		
	Handle parm. format=value:			
	If last parm. set expected delim. value to blankmelse to ','. Check for valid value and expected delimiter .>	PARS		
	If unexpected delimiter, branch.....>	SC90		
	Handle parm. format=(...)			
PANA	Set expected delim. to ',' ,BAL .....>	PARS		
	If delim. was ',', branch.....>	PANB		
	If delim. was not ')', branch.....>	SC90		
	Handle parm. format=(value)			
	Set expected delim. value ')', BAL..>	PARS		
	If error return, branch.....>	SCSD		
	Else branch.....>	PANX		
	Handle parm. format=(value,value)			
PANB	Set expected delim. value ')' and point R4 to the next subparm value..>	PARS		
	If unexpected delim., branch.....>	SC90		
	If 2nd subparm (userid) is in remid format (nnn), then convert to 'Rnnn'			
PANX	Return to caller		R5	
	Subparameter Checking Subroutine			
PARS	If no delim. was found in or after the subparm., branch.....>	PARL		
	If the found delim. does not equal the expected value, branch.....>	PARB		
PAR3	If the length of the subparm. is zero then accept default value, branch...>	PAR4		
	Otherwise move subparm. to output field addressed by R4.			
PAR6	Set return code= no error		R2	
PART	Return to caller		R14	
	Handle unexpected delim. condition:			
PAR8	If the unexpected delim. was not a '*', then branch with return code...>	PART	R2	
	If the '*' is not the first character then return with return code.....>	PART		
	If the expected delim. does not directly follow the '*', branch.....>	PART		
	Otherwise branch.....>	PAR6		
	Handle no delim. found condition:			
PARL	If the parm. was greater than 8 chars in length, branch.....>	SC90		
	If this was not the last parm. then return with return code in R2.....>	PART	R2	
	Otherwise, branch.....>	PAR3		

Labels	Chart SC: IPW$$SC - Scan Reader JECL Statement		Modified Data Fields	Reg. Usage	Calls
	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			
	16: ',', branch to continue scan ..>	SC80			
SC84	Register 1 is set to point to end of statement.			R1	
	Last parameter switch is set ON.		LWPI(IPW$DTC)		
SC86	Register 1 is saved in register 9.			R9	
	Branch to flag invalid statement....>	SC90			
SC90	Error exit: address of parameter in error is loaded in register 0 and made negative to signal error.			R0	
	Register 1 is set to next delimiter, as saved in register 9.			R9	
	Branch to exit.....>	SC94			
SC92	Normal exit: parameter address in register 7 is loaded in register 0.			R0	
	Address of next delimiter is loaded in register 1.			R1	
SC94	Return parameter registers are stored in save area.		SCR0(IPW$DSV)		
	Control is passed back to the caller.				IPW$REI

IPW$$SF - SAVE ACCOUNT FUNCTION FOR FBA DEVICES



HIPO-DIAGRAM SF-0.0 DATE - 10/27/81

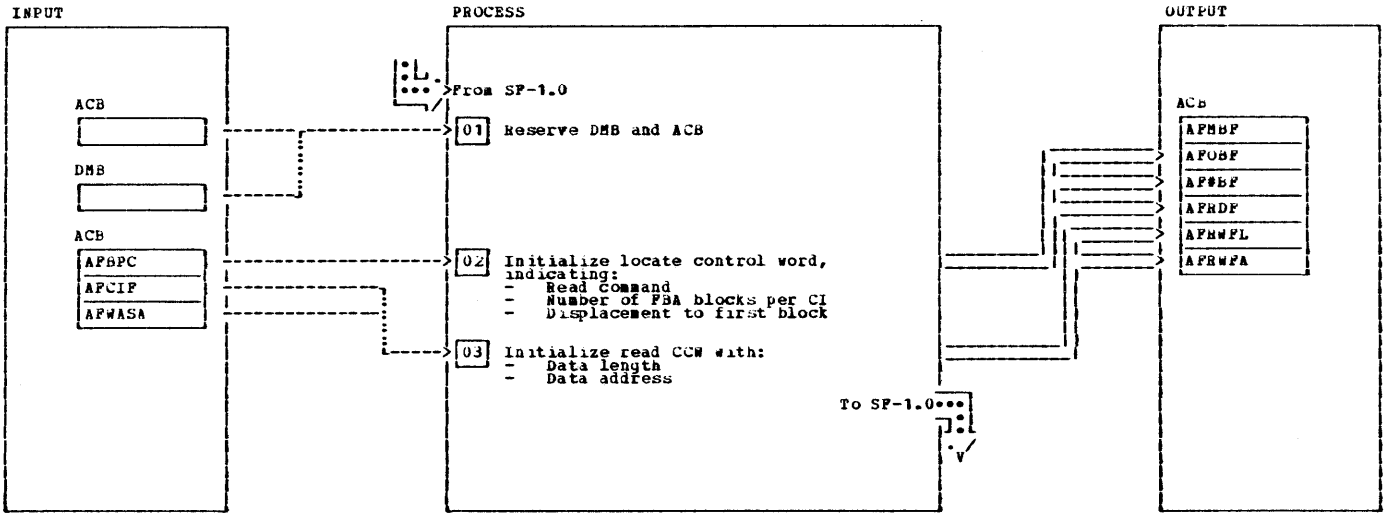

IPW$$SF - Mainline
INPUT
PROCESS
OUTPUT
From VSE/POWER Task Selection
01 Initialize ACB.
SFINIT
SF-2.0
02 If delete of Account file was requested:
IPW$$GF
A. Inform operator that Account file has been deleted.
05
TCB
ACB
Account File
03 Check for empty Account File.
SFEMPTY
SF-3.0
A. If Account File is empty...
05
04 Process PACCOUNT command.
A. If save to tape:
SFTAPE
SF-4.0
B. If save to disk:
SFDISK
SF-5.0
C. If save to card:
SFCARD
SF-6.0
ACB
05 Post Put Account task, if any is waiting.
Detach task
AFESF
AFTCB
Exit to VSE/POWER Task Selection

IPW$$SF - Mainline

	NOTES	MODULE	LABEL	REF
	The Save Account task is created when the central operator has issued a PACCOUNT command. Information about the save medium is passed by the command processor. The medium may be disk, tape, or the VSE/POWER punch queue. Also a delete of the VSE/POWER Account file is possible, which causes the Account file to be erased and an EOF-control interval to be written as first control interval on the file (to indicate empty data set).			
1	The account control block is set up with all information to read the VSE/POWER Account File.		SFSTART	
4	If an ERASE request is issued, the content of the Account File is not saved. The IPW$CAF function is invoked to write a Software End of File (SEOF) control interval to the VSE/POWER Account File. The account control block is set up to reflect the empty state of the Account File and the following message is issued:		SFDEL	$CAF
	1Q80I ACCOUNT FILE ERASED			$GAM
3	A check is made if any account records are on the Account File. If none are found, a message indicating that the file is empty is issued and the save function terminates.			
4	The content of the Account File is saved to a user-specified medium: tape (nolabel or standard label), FBA device, or punch cards. Then the Account File is initialized to empty state, and the account control block is set up to reflect the Account File.		SFSAVE	

	NOTES	MODULE	LABEL	REF
4A	The 'save-Account-file-to-tape' routine is invoked when the 'TAPE' parameters were specified in the PACCOUNT command. Its purpose is to read all control intervals from the Account file, deblock it and the write all account records to tape.			
4B	The 'save-Account-file-to-disk' routine is invoked when the 'DISK' parameter was specified in the PACCOUNT command. Its purpose is to copy all control intervals from the Account file to a user specified disk extent of the same device type.			
4C	The 'save-Account-file-to-card' routine is invoked when the 'PUN' parameter was specified in the PACCOUNT command. Its purpose is to read all control intervals from the Account file, unblock the account file, create punched cards for each account-record, and spool them to the VSE/POWER punch queue.			
5	The soft wait ECB on which either task might be waiting on is posted in order to allow it to resume processing. The TCB address is cleared to allow other PACCOUNT commands to be processed.			$DET

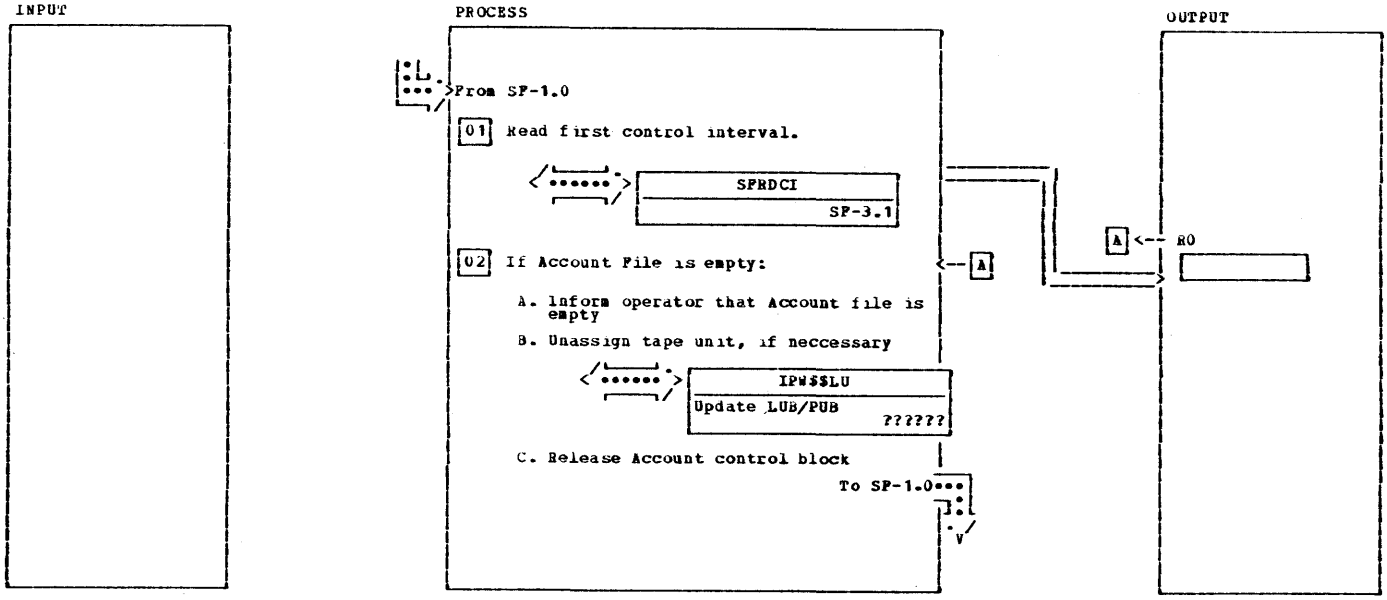
IPW$$SF - SFINIT Initialize ACB to Read Account File



NOTES	MODULE	LABEL	REF
1 The disk management block (DMB) and the Account control block (ACB) are reserved for exclusive usage in order to prevent simultaneous processing with other tasks that are writing to the account file. The account-function trace byte in the TCB is set to '0', to indicate that save account is being initialized The current block number of the Account file is saved in case an error occurs and the old status of the Account file has to be restored		SFINIT	$RSR

NOTES	MODULE	LABEL	REF
2 The extent description block, locate control word, CCB and CCWs, located in the account control block, are used to read the VSE/POWER Account File. 2 The locate control word is set up with following values: - number of FBA blocks/control interval - relative block number of first control interval (always 0) - indication that read command follows 3 The account-function trace byte is set to 'A', to indicate that 'open' of account file is completed			

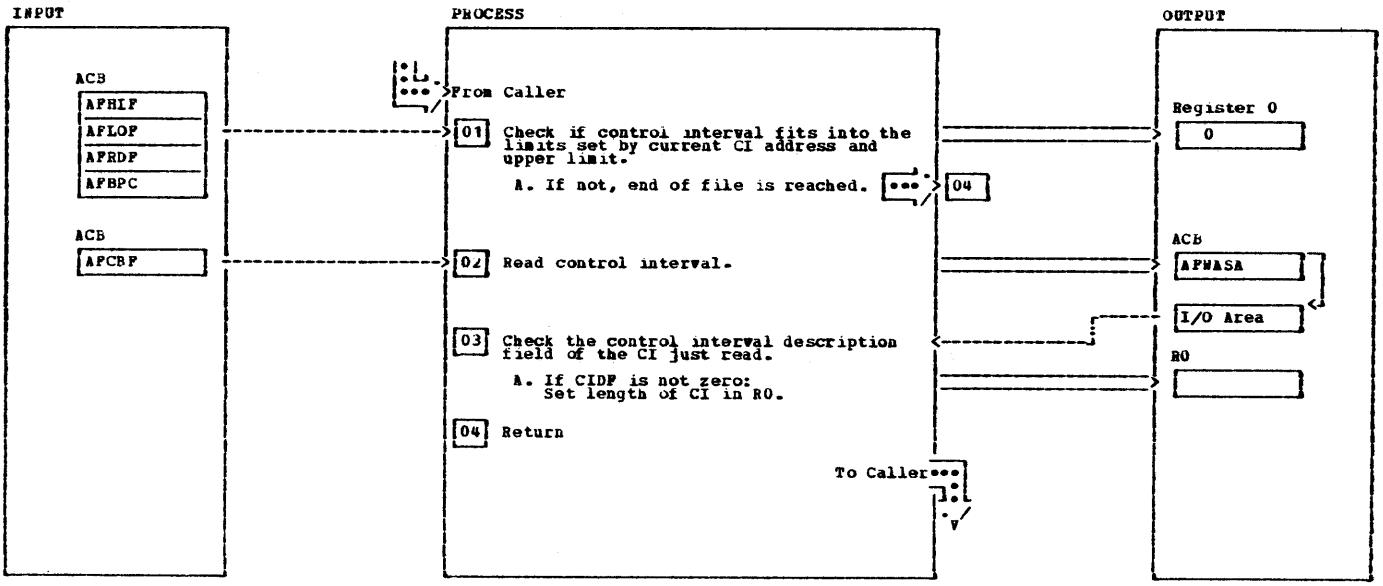
IPw$$SF - SFEMPTY Read First Control Interval to Check for Empty Account File



	NOTES	MODULE	LABEL	REF
1	The account control block was initialized by the SFINIT routine to allow the first control interval of the Account File to be read in. The control interval is checked by SFRDCI if it is empty or if it contains account data.		SFEMPTY	
2	Register 0 is returned with the length of the control interval, or with zeros if EOF was found.			
2A	If a Save Account task was started, but the Account File is empty, the following message is issued: 1083I ACCOUNT FILE NOTHING TO SAVE			$GAM

	NOTES	MODULE	LABEL	REF
2B	If the operator specified a tape as save medium, the tape unit is now unassigned.			$ULP
2C	The Account control block is released and the Account file is left unchanged. Return is made with a displacement to exit the save account function.			$RLB

IPW$$SP - SFRDCI Get a Control Interval from the Account File

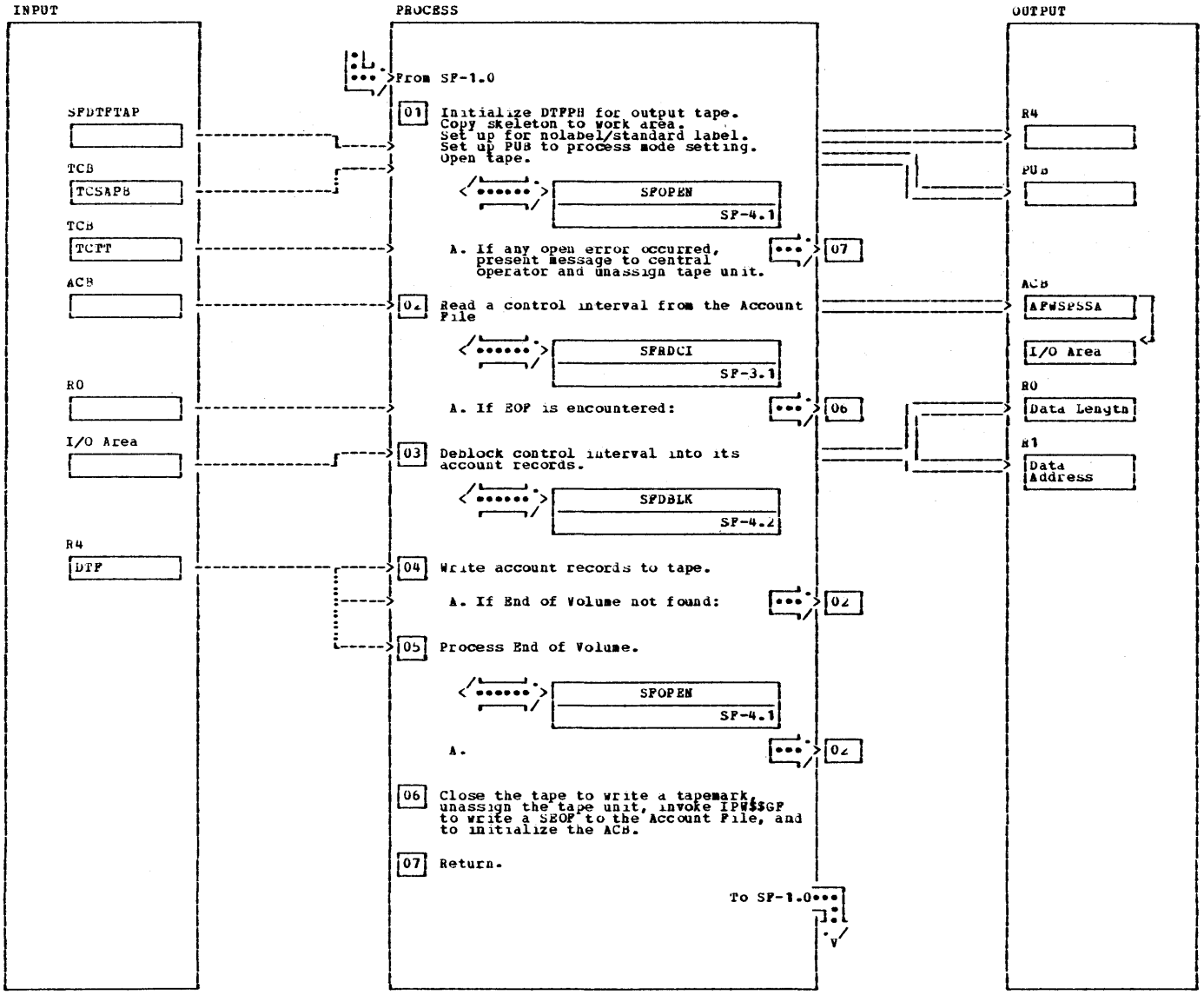


NOTES	MODULE	LABEL	REF
1 First the account-function trace byte in the TCB is set to signal 'get-account-record-active'. If no control interval can be obtained, an EOF condition is signalled to the caller by setting Register 0 to binary zero.			
The start address of the control interval on the Account File is passed by the caller. A check is made if the CI can be read, or if it would exceed the file's upper limit.		SFRDCI	
2 If a control interval can be obtained, it is read into the workarea provided by VSE/POWER initialization in the GETVIS area			
A DOS/VSE EXCP and VSE/POWER IPW$WFC request are issued. The CCB, CCWs, and control blocks located in the account control block are used to perform the I/O operation.	EXCP		$WFC

NOTES	MODULE	LABEL	REF
3 The control-interval description field (CIDF) of the control interval just read is analyzed. If the CIDF contains all zeros, this control interval is the Software-end-of-file(SEOF) condition for the Account file, in which case Register 0 is set to zero, to signal EOF to the caller.			
3A If the CIDF indicates a data-control interval , which means the CIDF contains not all zero, Register 0 is set up with the length of the control interval.			

This page was left blank intentionally.

IPW$$SF - SFTAPE Save Content of the Account File to Tape

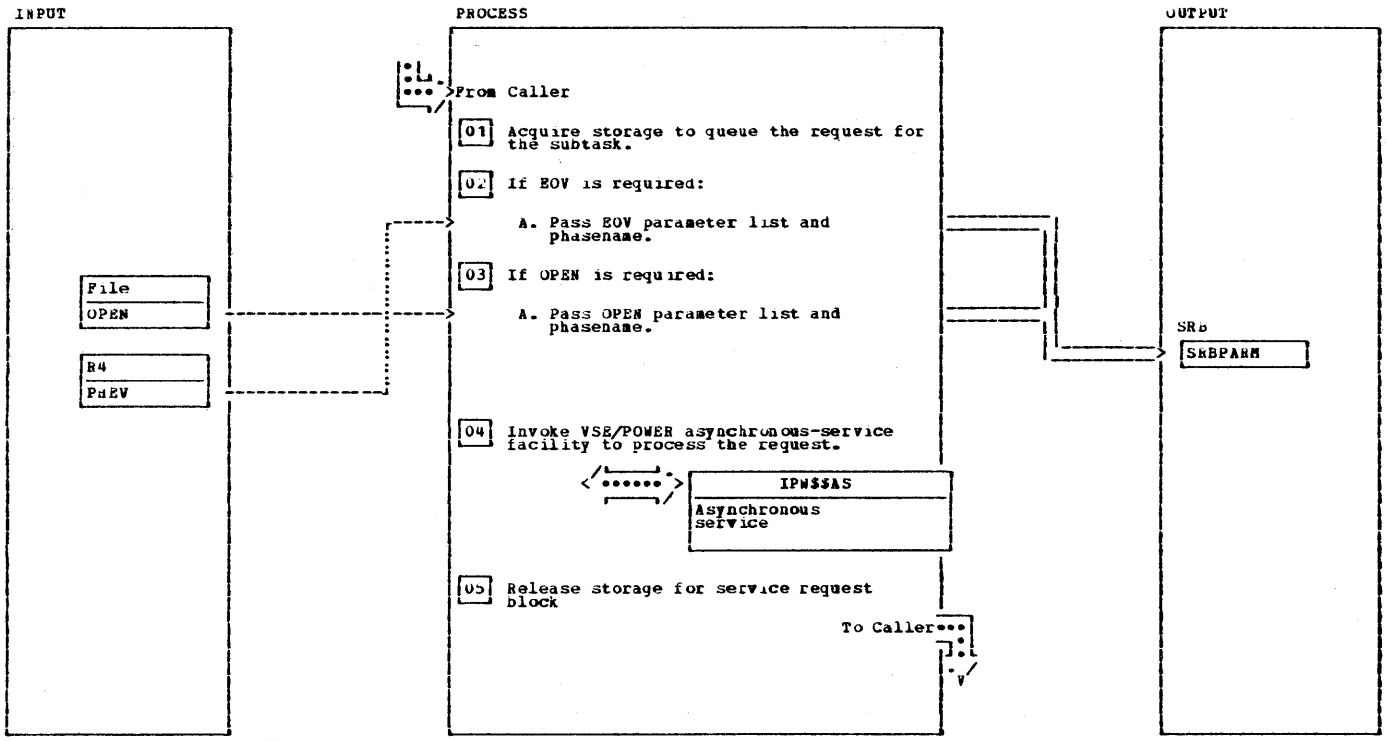


IPWSSSF - SFTAPE Save Content of the Account File to Tape

NOTES	MODULE	LABEL	REF
1 The assembled DTF skeleton is copied into a work area. The DTF copy is then initialized with the tape information from the command processor. As nolabel and standard-label tapes can be processed, the DTF is set up accordingly. The user-specified tape density is set to the PUB.		SFTAPE	$RSW
The VSE/POWER asynchronous-service facility is invoked to establish a DOS/VSE subtask to process the open request. A DOS/VSE subtask is used to prevent VSE/POWER cancellation caused by label-type processing errors.		SFTAPE2	
2 A control interval may contain one or more account records. One control interval is read, and EOF is signalled by SFRDCI routine to the caller by setting R0 to zero.			
3 The control interval is deblocked into its account records by SFDBLK routine, and each account record is written sequentially to tape. When all account records for a control interval are processed, the next one is obtained.	EXCP		$WFC

NOTES	MODULE	LABEL	REF
5 The VSE/POWER asynchronous-service facility is invoked to establish a DOS/VSE subtask to process the EOV request.		SFTAPE7	
6 Subroutines SFUNASS and SFWTO are invoked to unassign the tape unit and to issue this message:		SFTAPEEN	$FCH
1Q79I Account FILE SAVED			$CAF
			$CAF
7 Issue message if no storage could be obtained to copy the DTF:		SFTAPEE2	$GAM
1Q78I NO REAL/FFIXED STORAGE AVAILABLE			
Issue this message if an error was encountered during open or EOV processing of the output tape:			
1Q60I OPEN FAILURE ON PAccount OUTPUT DEVICE			

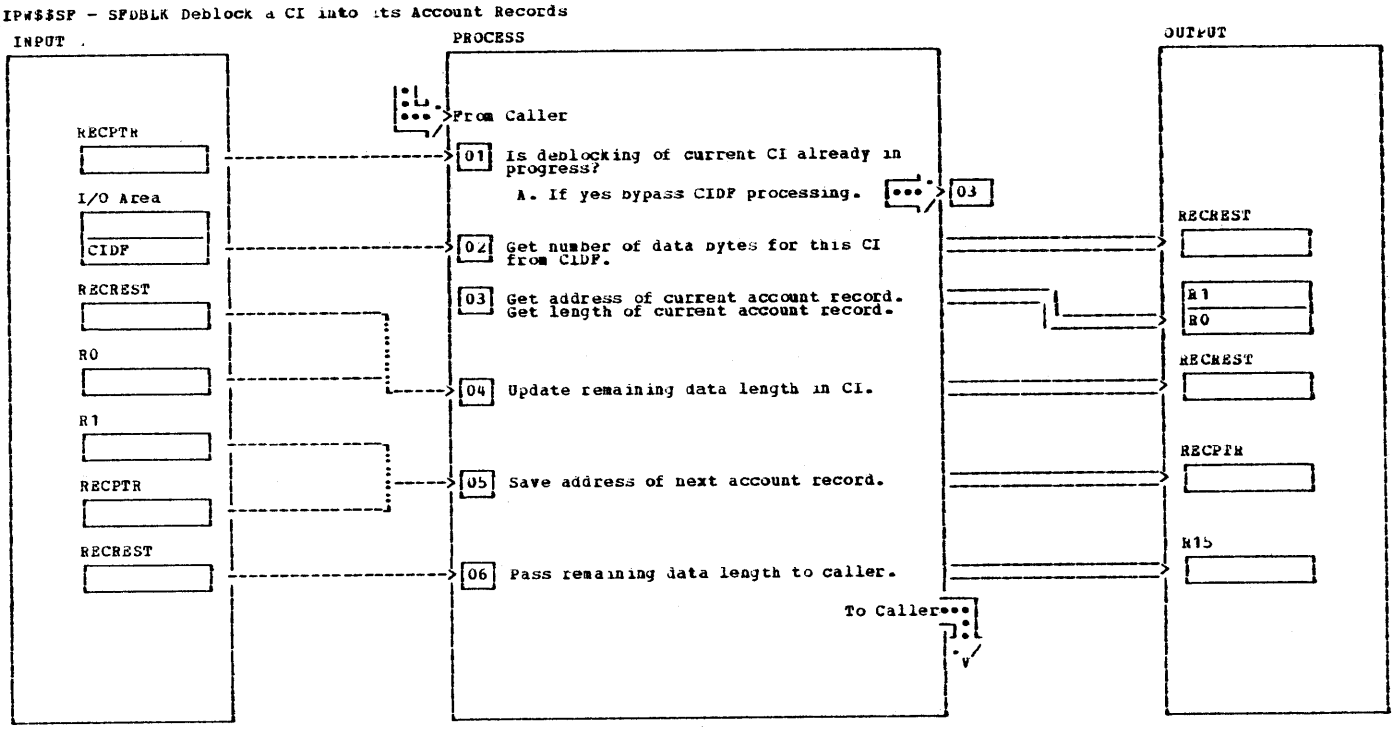
IPW$$SF - SFOPEN Open/EOV Processing for Save Medium on FBA Device or Tape



NOTES	MODULE	LABEL	REF
A DOS/VSE subtask is used to process the open request fo the user's FBA device and tape, and to process EOV for the tape. As OPEN modules for FBA are located in the SVA, page faults during OPEN time would be intercepted by VSE/POWER which in turn cannot handle them. To avoid these problems, a DOS/VSE subtask is used. Operator handling errors during OPEN/EOV time of labeled tape can cause VSE/POWER to be cancelled. Using a subtask will cause the subtask to be cancelled in this case, while the VSE/POWER maintask still can continue.			

NOTES	MODULE	LABEL	REF
1 Storage for the service request block is acquired and formatted. Indicate transient fetch requested.		SFOPEN	#SW
4 Asynchronous service is called to handle the request. The task is put is wait state until the request is completed.			$IAS
5 The storage is returned to the VSE/POWER storage pool.			$#LW

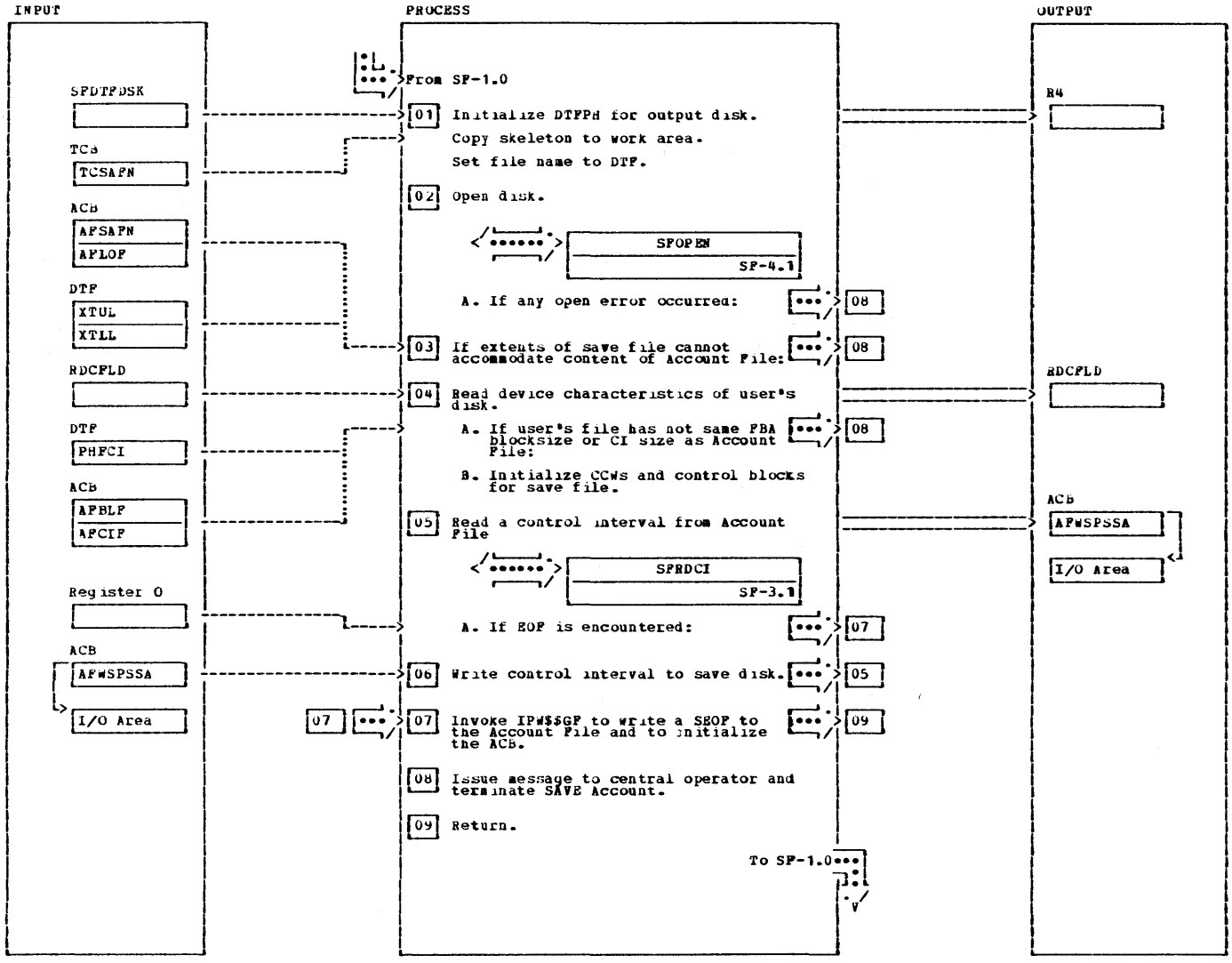
IPW$$SF - SFDBLK Deblock a CI into its Account Records



NOTES	MODULE	LABEL	REF
1 If field RECPTR is zero, a new CI has to be deblocked.		SFDBLK	
2 The remaining data length for the current CI is saved in RECREST and is decremented each time an account record is passed to the caller.			
3 The length and address of the account record are passed to the caller in registers 0 and 1.		SFDBLK1	
5 The pointer to the next block behind the one in process is saved for late use when the deblock routine is entered again for this control interval. A check is made to determine if the current block is the last one for the control interval in process. If it is the last one, 'RECPTR' is set to zero to indicate to the deblock routine that a new control interval is in process (the next time the routine is entered).			

NOTES	MODULE	LABEL	REF
6 R15 contains the remaining data length within the current CI. If the last account record was retrieved from the current CI, R15 is set to zero to indicate that the CI is processed. Upon return, the caller may check R15 to acquire a new control interval, if R15 is zero. If the save device is a tape, the address and the length of the account record including its header are passed to the calling routine. If the save device is card image for the punch queue, the header is stripped off and the address and length of the account record without header are passed to the calling routine			

IPW$$SF - SFDISK Save Content of the Account File to FBA Device

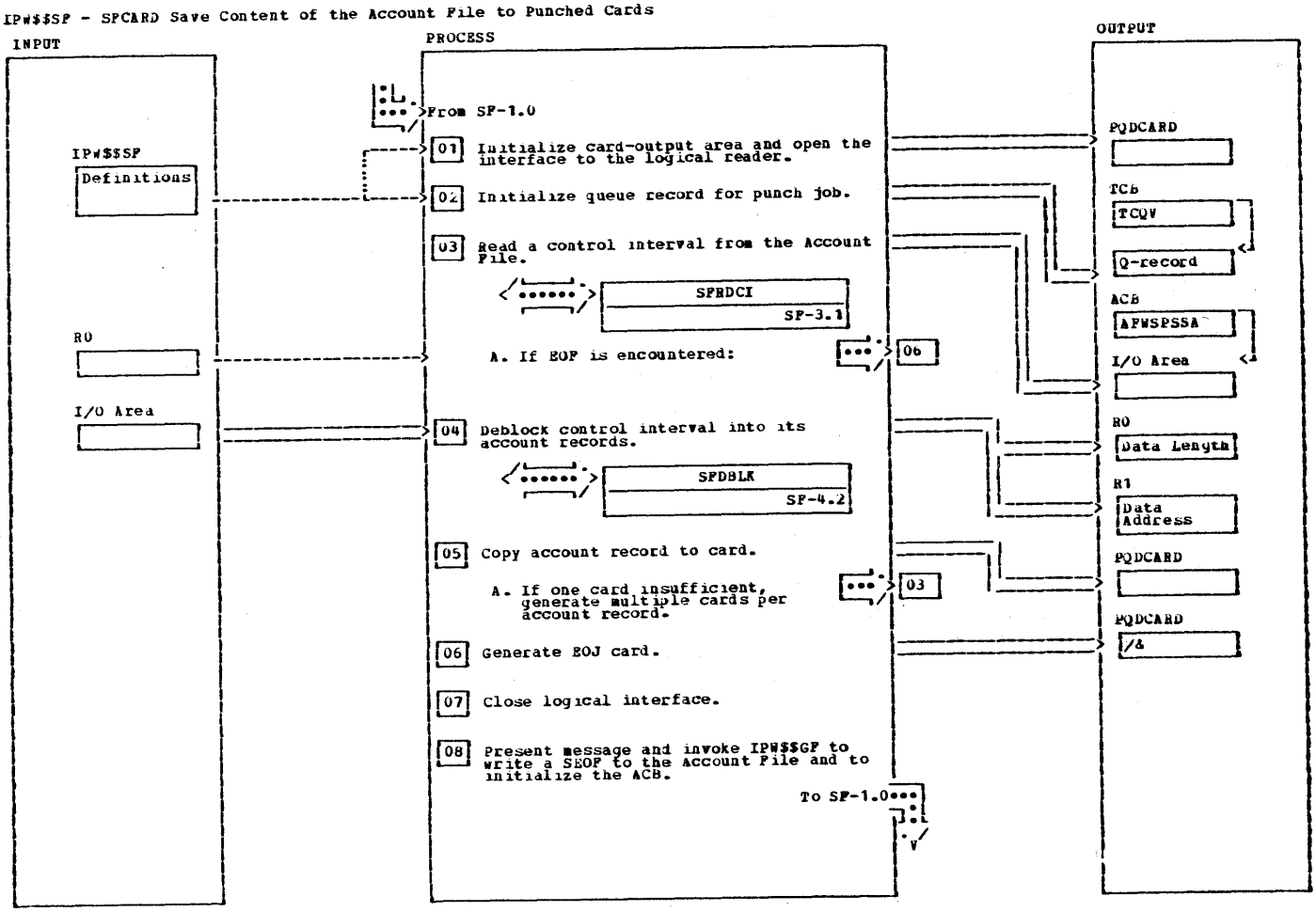


IPW$$SF - SFDISK Save Content of the Account File to FBA Device

	NOTES	MODULE	LABEL	REF
1	The assembled DTF skeleton is copied into a work area. The DTF copy is then initialized with the disk information from the command processor. The file name for the save disk was supplied with the PACCOUNT command.		SFDISK	$RSW
2	The VSE/POWER asynchronous-service facility is invoked to establish a DOS/VSE subtask to process the OPE request. A DOS/VSE subtask is used to handle the page faults of th OPEN modules, located in the SVA.			
3	The number of occupied FBA blocks on the Account File is calculated and checked against the extent of the save file. As only control intervals containing account data are copied, the save file may be smaller than the Account File, if the latter is only filled partially. If the save file is too small, the following message is issued: 1Q81I filename EXTENT TOO SMALL, COMMAND NOT EXECUTED.			
4	A Read-Device-Characteristics CCW is issued to get the FBA blocksize of the save file. This block size has to match the one of the Account File. If no match, the save-account function cannot be executed and the following message is issued: 1Q3CI INVALID BLOCKSIZE FOR filename A check is made for specification of a CI size parameter for the save file. It is recommended that this parameter be omitted. If not, it has to match the CI size selected by VSE/POWER for the Account File. If no match, the save-account function cannot be executed and the following message is issued: 1Q3DI INVALID CI SIZE FOR filename.	EXCP		

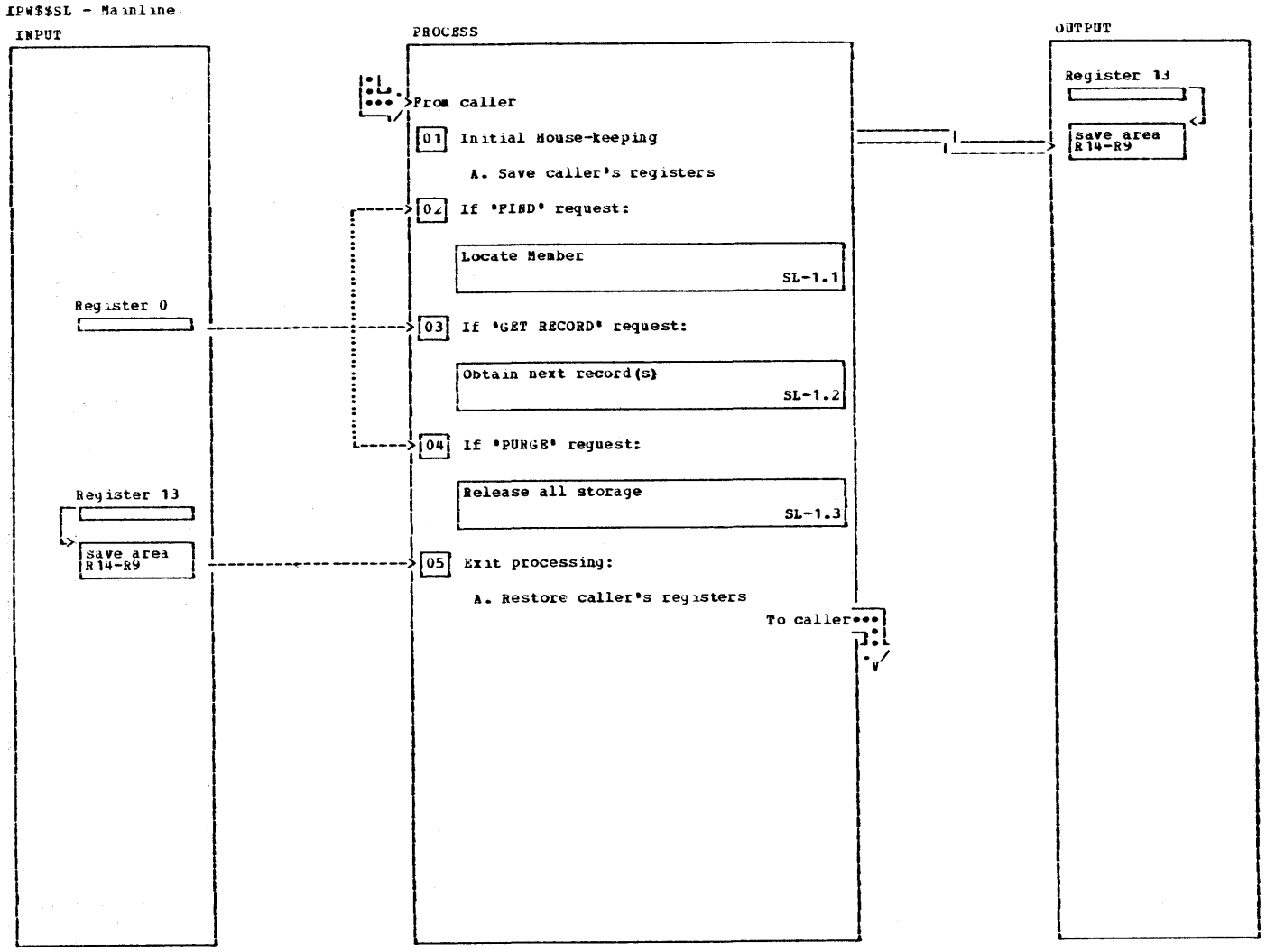
	NOTES	MODULE	LABEL	REF
5	The control intervals read from the Account File are written to the save file in the same CI; length and data are left unchanged. One control interval on the save file may contain one or more account records.	EXCP	SFDISKL	$WFC
6	The CCB part of the copied DTF and CCW's and control blocks located in the IPW$$SF module are used to write the save file.	EXCP —		$WFC
7	Issue message: 1Q79I ACCOUNT FILE SAVED		SFDISKEN	$CAF $RLW $GAM

IPW$$SF - SPCARD Save Content of the Account File to Punched Cards



	NOTES	MODULE	LABEL	REF
1	Initialize card sequence to assign a unique card sequence to each account record, accompanied by an ascending number for each card of a multiple card-account record.		SPCARD	$OLI
2	The queue record is initialized for a punch job with job name, class, priority, and disposition.			
3	Invoking SFRDCI routine, each control interval is read from the Account File until EOF is hit.		SPCARD1	
4	The control interval is deblocked into its account records, and each account record is punched to card. If one account record does not fit on one card, multiple cards are punched for this record.		SPCARD2	

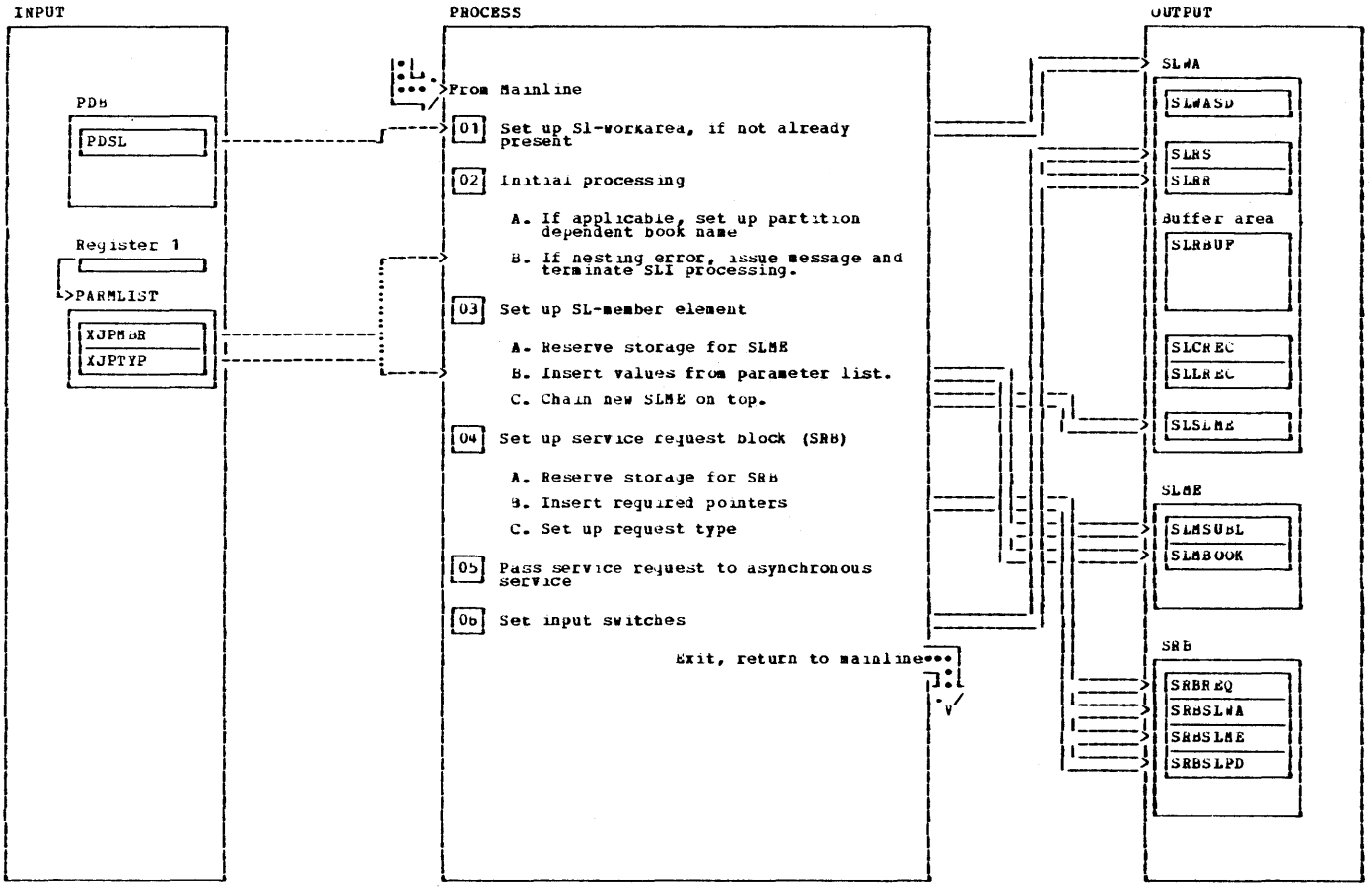
	NOTES	MODULE	LABEL	REF
5	An EOJ card is generated to close the card stream. Each card has an eight-byte sequence number in positions 73-80. Positions 73-78 form an ascending number for each account record while positions 79 and 78 are used to increment the card number if a multiple-card record is processed.		SPCARDEN	$CLI $CAF



NOTES	MODULE	LABEL	REF
1 The caller's registers are saved in the save area addressed by register 13.			$SAV

NOTES	MODULE	LABEL	REF
5 The caller's registers are restored from the save area addressed by register 13 and return is made to the caller			$RET

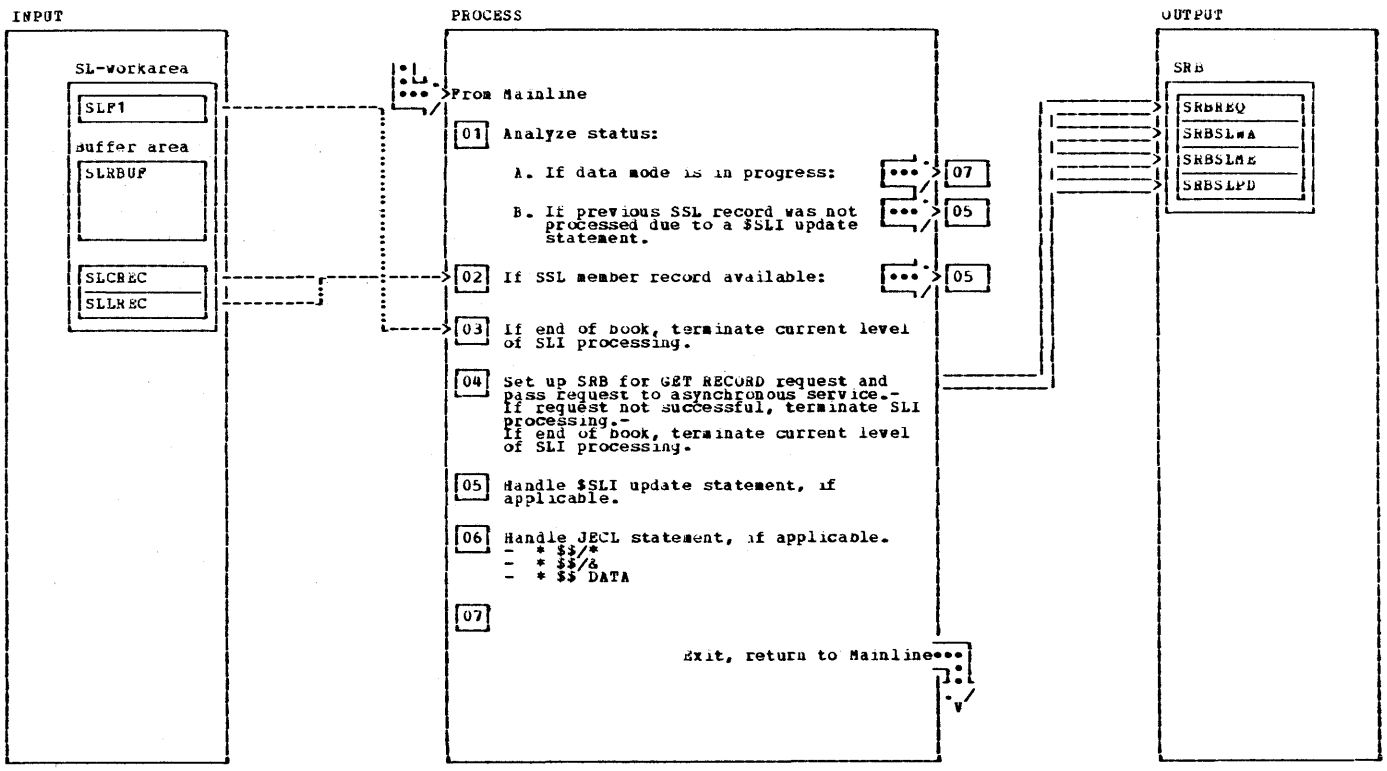
IPW$$SL - Process FIND request



NOTES	MODULE	LABEL	REF
1 If this is the first SLI request, storage for the SL-workarea (SLWA) is reserved and its address is stored in the partition control block (PDB) of the related task. The storage descriptor for the SLWA is set up. The SLWA contains among others the buffer area for ten 80-byte logical member records and address fields for pointers to the current and last record within the buffer area.		SLFIND	$RSV
2A If the first two characters of the name area are '$$', the second character is changed according to the id of the partition being serviced. (0 for BG, 1 for F1, ...)		SLF05	
2B The SLME chain is scanned to locate any SLME addressing the same member. If found, a nesting error occured. Message 1R36I 'SLI NESTING ERROR, BOOK bookname is issued and SLI processing is terminated by entering the PURGE function.		SLF07	$GAM
3 A SL member element (SLME) is used to describe a member. It contains note/point information, bookname, sublibrary and others. A SLME exists for each source statement library member currently in the insertion process.		SLF35	
3A If no storage could be obtained, SLI processing is terminated by entering the PURGE function.			$RSV
3B Sublibrary and bookname are copied from the parameter list pointed to by register 1 at function entry.			

NOTES	MODULE	LABEL	REF
3C The just formatted SLME is queued as first entry in the SLME stack. The current SLME pointer is updated to point to the new SLME.		SLF45	
4A Fixed storage is acquired for the SRB because it contains an ECB, which must be testable without the occurance of a page fault.			$RSW
4B The pointers to the SL-workarea, the SL member element and the partition control block are stored into the SRB.		SLF50	
4C The 'FIND MEMBER' request type is inserted.			
5 Register 1 is loaded with the address of the service request block and asynchronous service is called. If the request is not successful, SLI processing is terminated by entering the PURGE function.			$IAS
6 On the highest level the read reader switch is set to indicate that a record from the reader input queue must be read as next in order to process * $$ DATA and $SLI update statements. The read SSL switch is set off because the 'FIND MEMBER' handling by the asynchronous service implies a read of the first member records, so that for a subsequent 'GET RECORD' request data are available.			

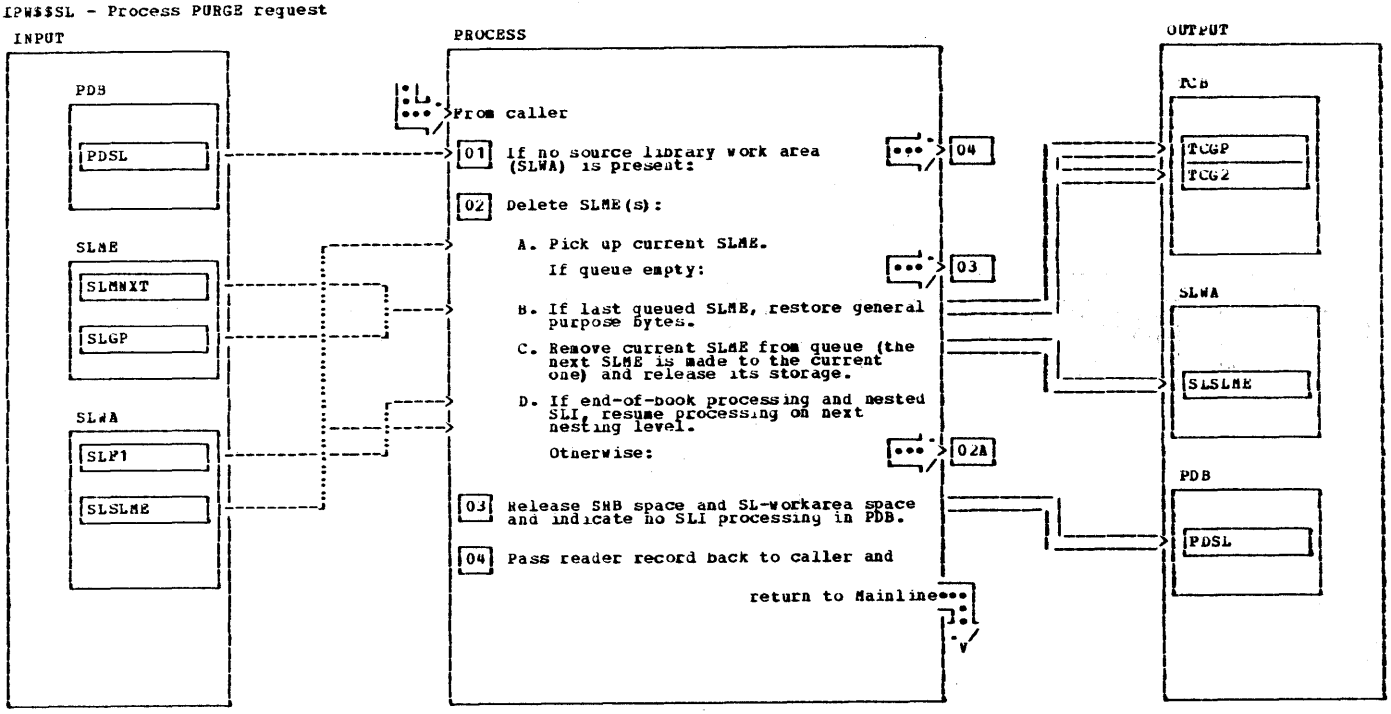
IPW$$SL - Process GET RECORD request



NOTES	MODULE	LABEL	REF
This routine is entered when a non first time request is made for a logical record from the SSL book. If data processing is in progress, exit will be done to signal data processing to the caller.		SLGETR	
1A If the last read in record is not a /*, /& or /*SLI statement, continue data mode processing.			
2 If the internal buffer area of the SLWA contains a next member record, this record is processed.		SLG20	
3 The current level of SLI processing is terminated by entering the PURGE funtion, which will resume processing on next nesting level in case of nested SLI or terminate in case of single SLI processing.			
4 The pointers to the SL-workarea, the SL member element and the partition control block are stored into the SRB. The GET RECORD request type is inserted. Register 1 is loaded with the address of the SRB and asynchronous service is called.- If end-of-book or an error is returned, SLI processing is terminated by entering the PURGE function, which checks again for end-of-book to control further processing.		SLG25	

NOTES	MODULE	LABEL	REF
5 If a $SLI update statement is present, a check is made wether the update statement must be passed to the caller. Following update functions are supported: - delete statement - insert after or before stmt - replace statemnt.		SLU05	
6 If VSE/POWER end of data record (* $$/*) make normal DOS/VSE end of data record out of it (/*). If VSE/POWER end of job record (* $$/&) make normal DOS/VSE end of job record out of it (/&).		SLJ05	

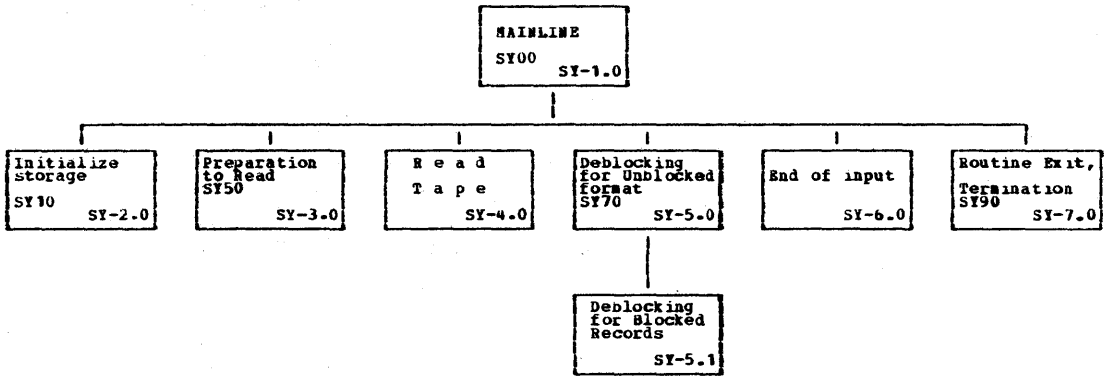
IPW$$SL - Process PURGE request

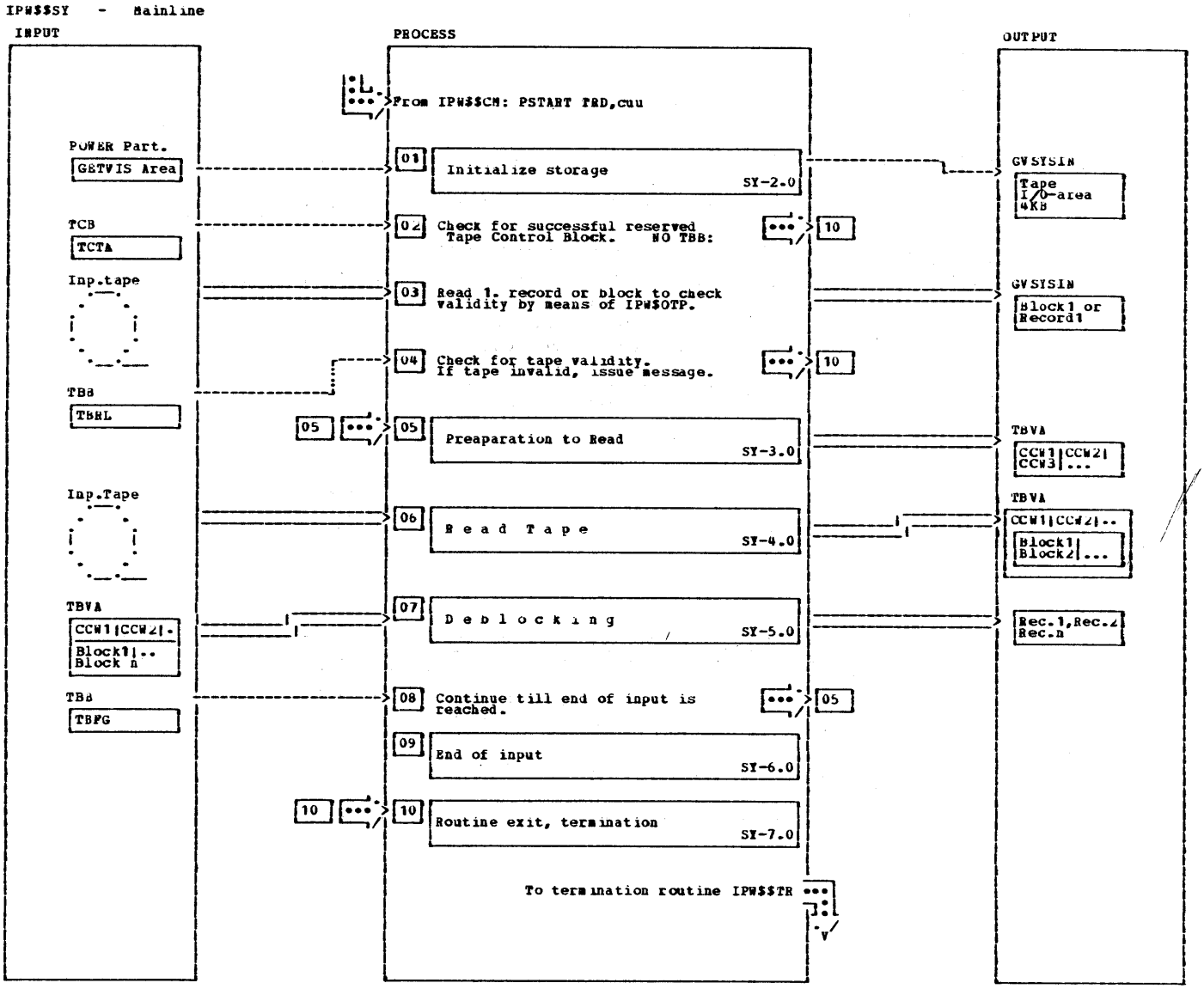


NOTES	MODULE	LABEL	REF
In case of nested SLI and end-of-book processing the current source library member element (SLME) is unchained and released and processing on the next nesting level is resumed.- Otherwise SLI processing is terminated by releasing all related work spaces and clearing the SLWA pointer in the partition control block.			
1 The SLWA pointer of the partition control block is examined. If it is present, work spaces must be released		SLPURGE	
2B The chain pointer of the just addressed SLME is examined. If there is no further SLME on chain, the general purpose bytes saved in the SLME are restored into the TCB.			

NOTES	MODULE	LABEL	REF
2D If end-of-book is flagged in the SLWA, the flag is reset and if there is a next SLME on chain (i.e. nested SLI), the GET RECORD function is reentered at asynchronous service request for next member record(s). Otherwise the deletion process is continued.			
3 The storage used for the SRB is released and returned to the VSE/POWER storage pool.		SLP1B	$RLW
The SL-workarea is released and the pointer to it in the partition control block is cleared.			$RLV

SYSIN TAPE READER TASK

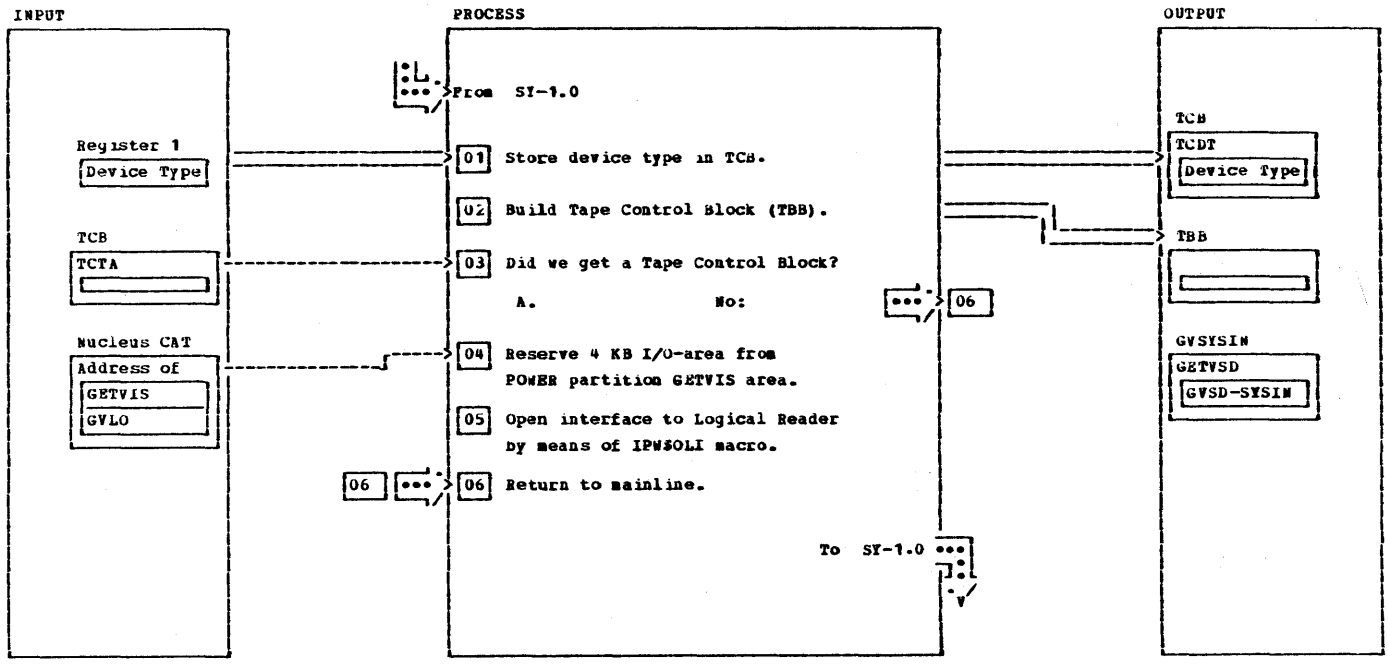




NOTES	MODULE	LABEL	REF
VSE/POWER provides full tape SYSIN reader support for fixed blocked and unblocked tape input. The maximum blocksize for blocked tape input is 4080 bytes. Record size must be 80 bytes fixed length, and blocksize must be a multiple of 80 bytes. For unblocked tape input fixed length records of 80 or 81 bytes are supported. Support is provided for IBM standard labelled tape files as well as unlabelled tape files.			
2 Tape Control Block (TBB) contains information referring to tape processing like cuu, I/O-area address, CCB, blocksize, queue-ID, etc.		SY00	

NOTES	MODULE	LABEL	REF
3 IPW$OTP-macro calls IPW$$OT-phase from where Read Tape Service routine of VSE/POWER Nucleus phase will be invoked. After every I/O-operation a check is made whether task has been stopped by operator. If yes, the SYSIN-tape-RDR task is terminated.		SY20	$OTP
4 If invalid record found, msg. 105AI INVALID TAPE MOUNTED FOR ttt,cuu is issued and task is stopped.			$WTO
8 End of input is indicated by 2 tape marks.			

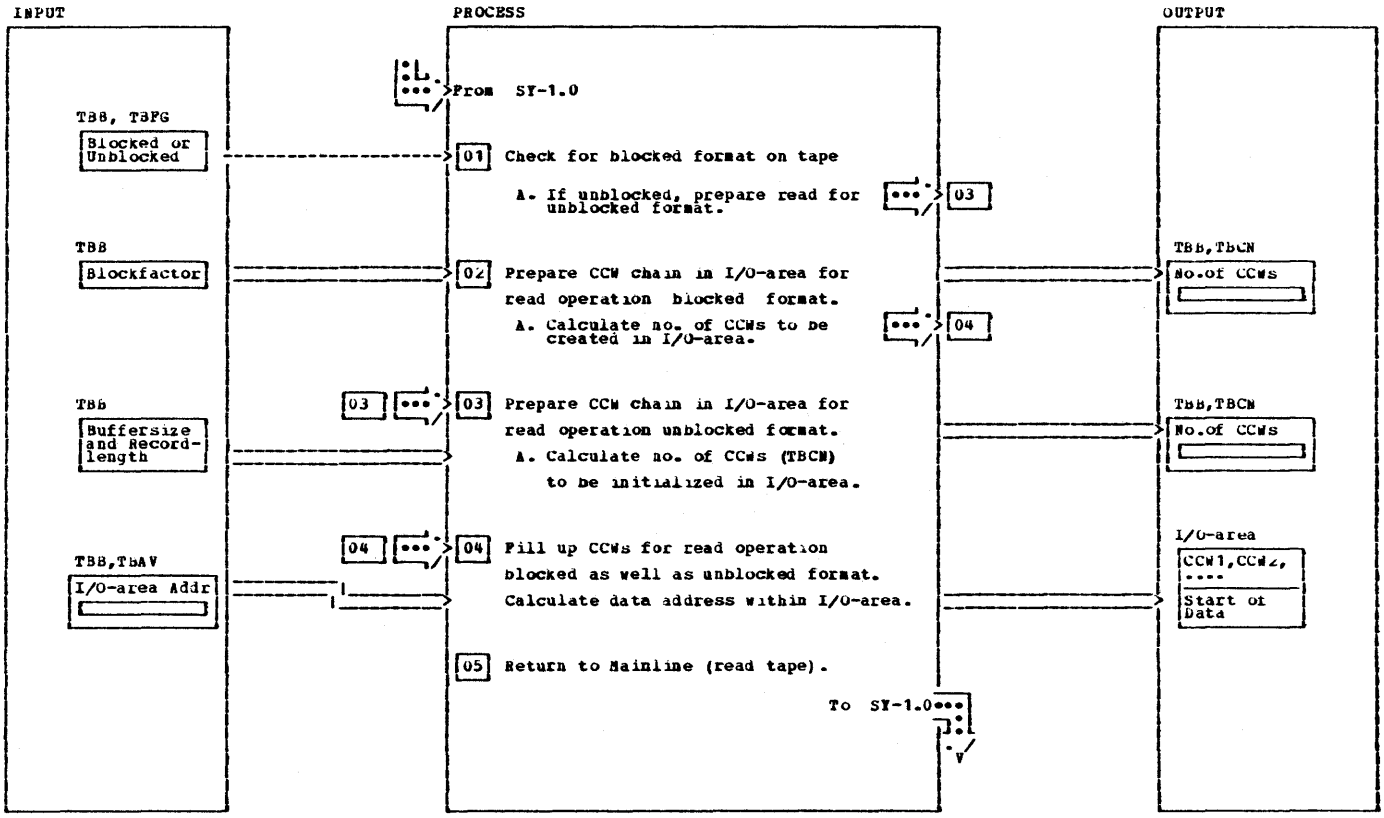
IPW$$SY - Initialize Storage



NOTES	MODULE	LABEL	REF
1 On entry at this point the device identifying information passed by the Command Processor in register 1 is stored in TCB.		SY10	
2 A Tape Control Block is built by invoking Open Tape Routine by means of IPW$OTP macro.			OTP
4 The 4Kb storage is reserved from the VSE/POWER partition GETVIS area.		SY06	$RSv

NOTES	MODULE	LABEL	REF
5 The logical reader scans each record fetched from I/O-area, performs any necessary JECL operations and passes the record to Data file. For the logical RDR the SYSIN RDR task is a normal physical RDR task.			$OLI

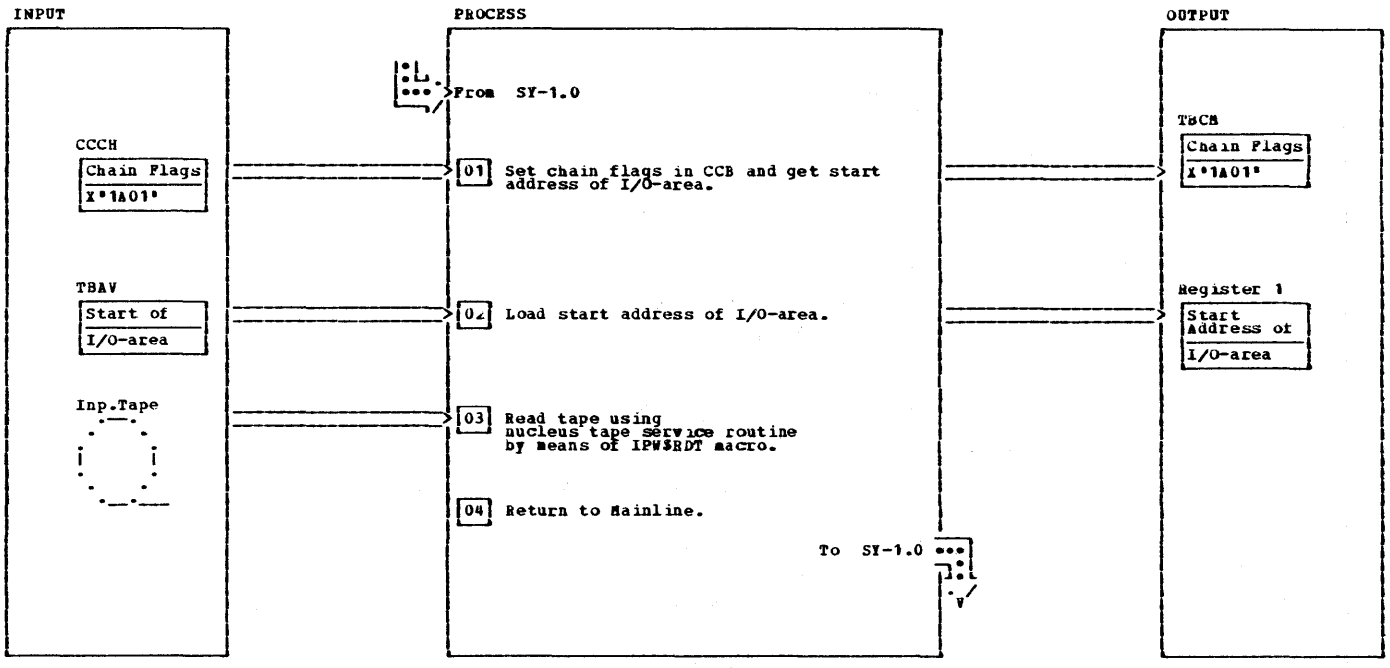
IPW$$SY - Preparation to Read



NOTES	MODULE	LABEL	REF
1 Depending on blocked or unblocked format of tape, a certain CCW in TBB and read-CCW is set up in I/O-area. R1 holds virt. address of I/O-area (workspace).		SY50	

NOTES	MODULE	LABEL	REF
4 Data addr.: TBRA= TBAV+(TBCN+8)+TBRL*TBCN TBRA= start addr. of data within I/O-area. TBAV= virt. start address of I/O-area. TBCN= No. of CCWs TBRL= Record length		SY54	

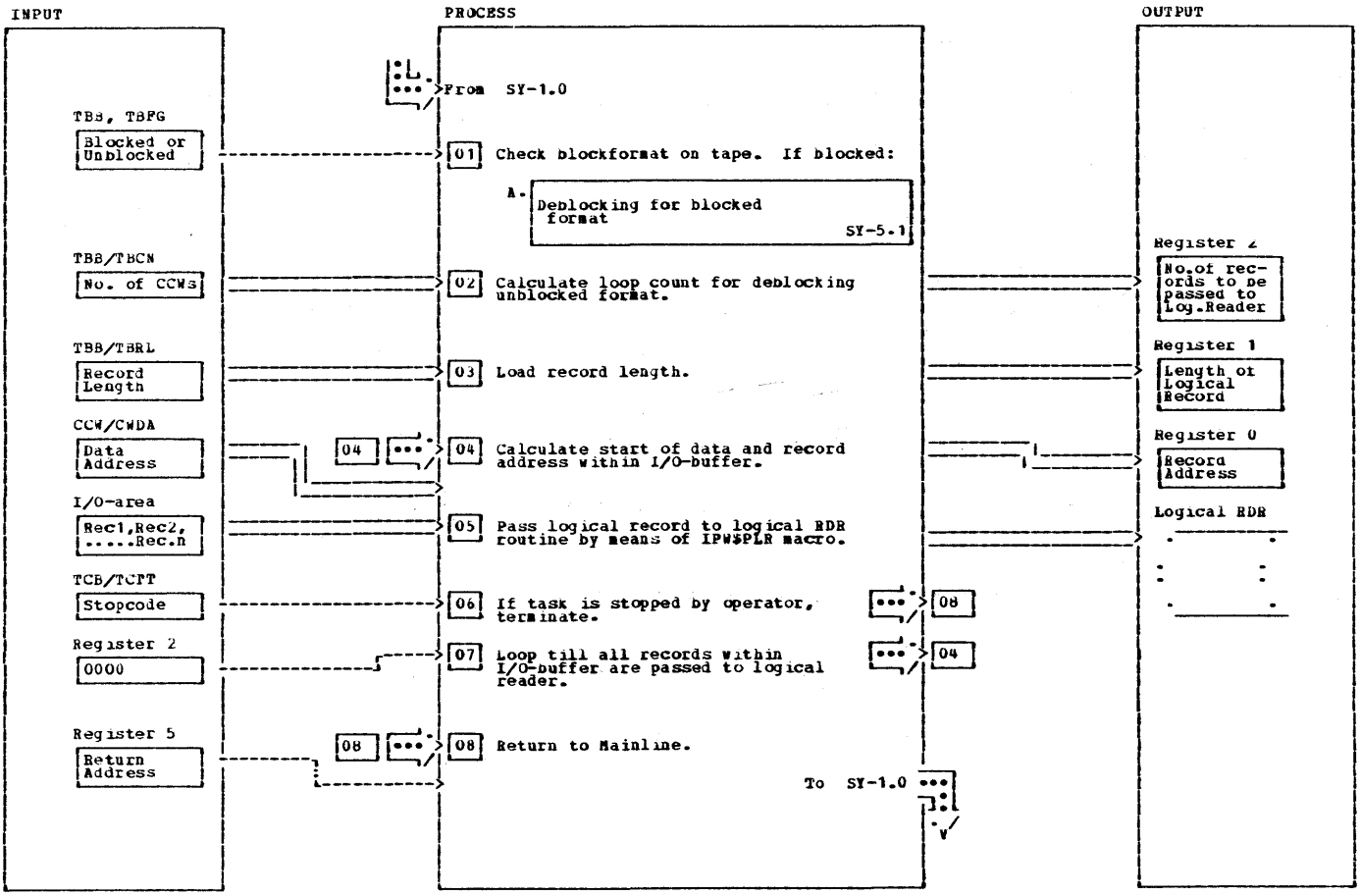
IPW$$SY - Read Tape



NOTES	MODULE	LABEL	REF
In both cases (blocked as well as unblocked format) the channel program in the I/O-buffer is executed and a max. number of logical records are read into I/O-area. The I/O-buffer is filled up to its maximum with only one read operation respectively.		SY55	$RDT

NOTES	MODULE	LABEL	REF
3 Immediate stop code 'S' in TCB is checked in mainline. Wrong length, unrecoverable I/O-error is checked and handled within tape service routine in nucleus. Read tape is completed now and control is given back to deblocking routine.			$RDT

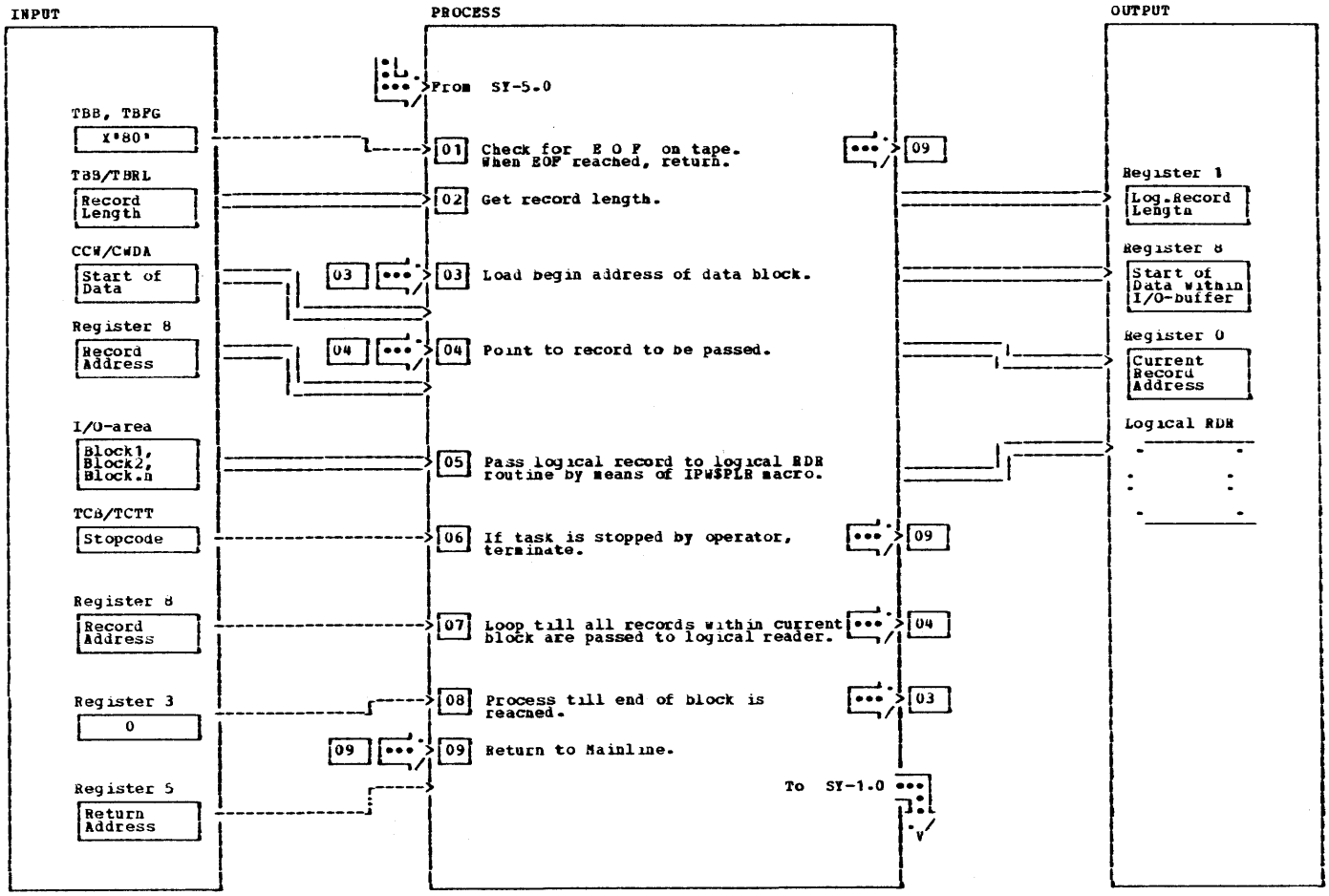
IPW$$SY - Deblocking for unblocked format



NOTES	MODULE	LABEL	REF
1 Each logical record in I/O-buffer is now passed to logical reader routine by means of IPW$PLR macro. Logical RDR invokes data management which passes records to Data file. Register usage within this subroutine: unblocked input: R0: Address of log. record R1: Length of log. record R2: Loop control register R6: Pointer to CCW currently in access.		SY70	$PLR

NOTES	MODULE	LABEL	REF
2 When EOF is found, loop count is set to zero and return is made to Mainline.			
5 Put logical record.			$PLR

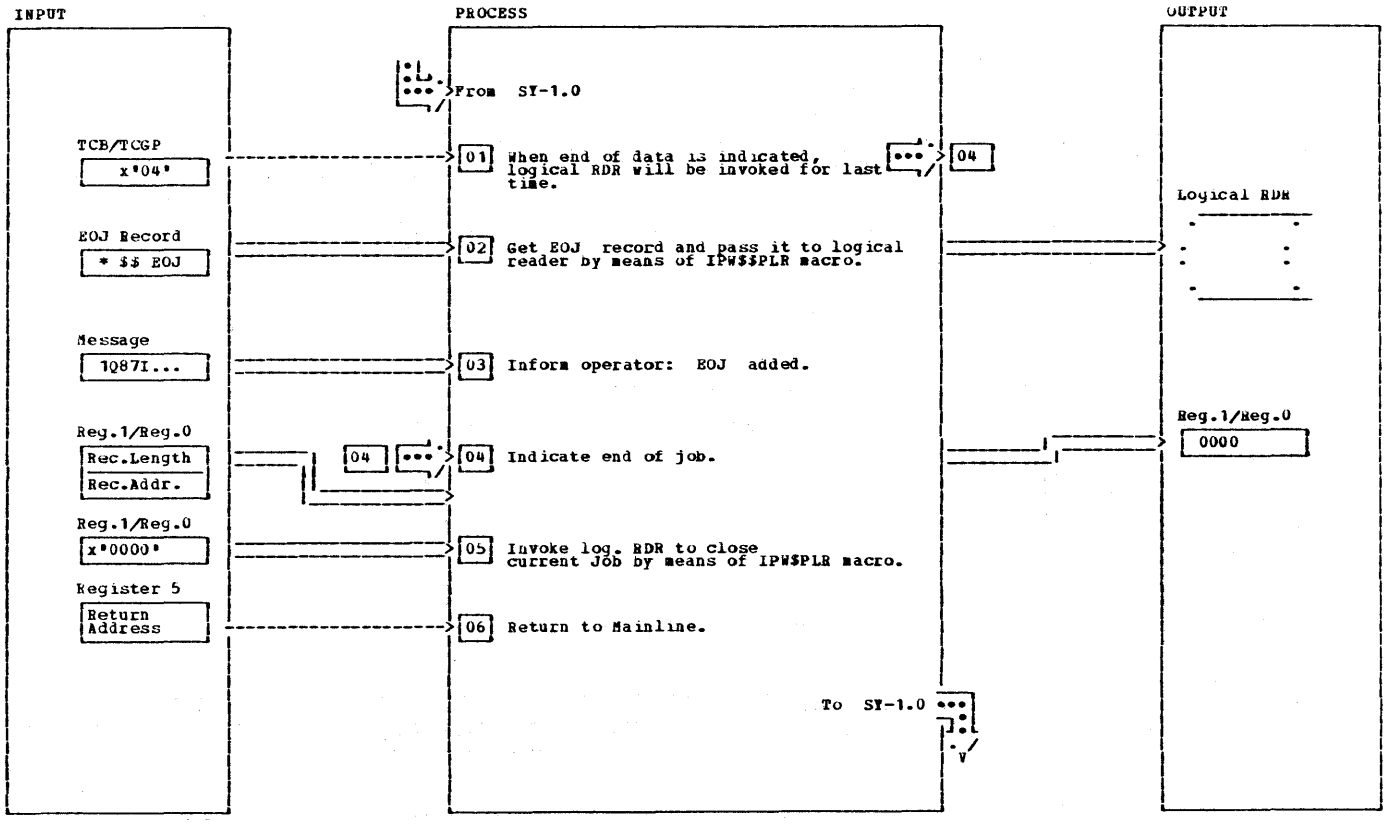
IPW$$SY - Deblocking for Blocked Format



NOTES	MODULE	LABEL	REF
2 Register usage within this subroutine: Blocked input: R0: Address of log. record R1: Length of log. record R2: Loop control register R3: Block counter R8: Address of record within input block.			

NOTES	MODULE	LABEL	REF
5 Each logical record in I/O-buffer is now passed to logical reader routine by means of IPW$PLR-macro. Logical RDR invokes data management which passes records to Data file.		SY75	$PLR

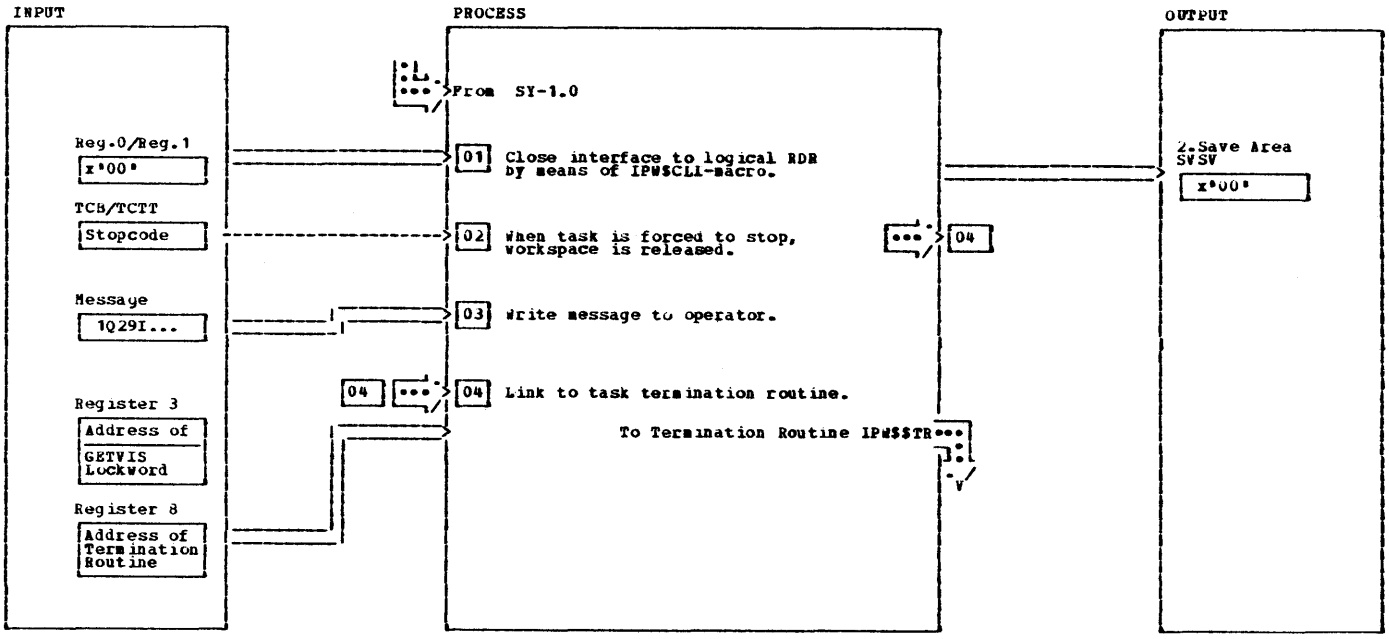
IPW$$SY - End of Input



NOTES	MODULE	LABEL	REF
3 Msg.: 1Q87I EOJ ADDED jobname jno is issued			$WTO

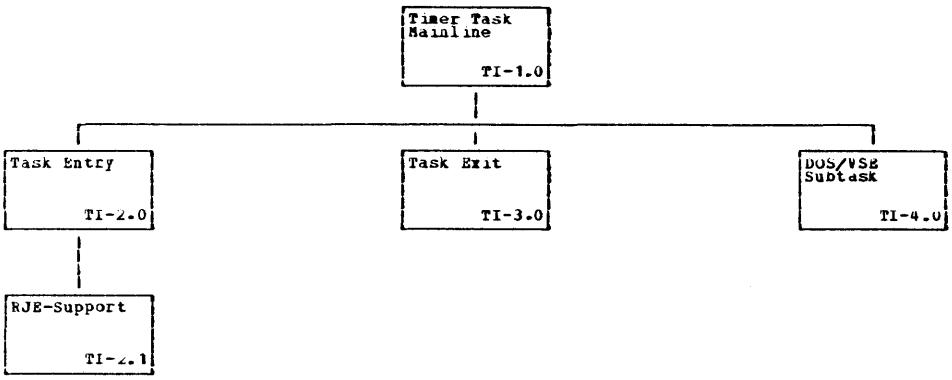
NOTES	MODULE	LABEL	REF
5 R1 = 0 and R0 = 0 indicates to logical RDR, that all records are passed and interface with data management can be closed.		SY7P	$PLR

IPW$$SY - Routine Exit, Termination

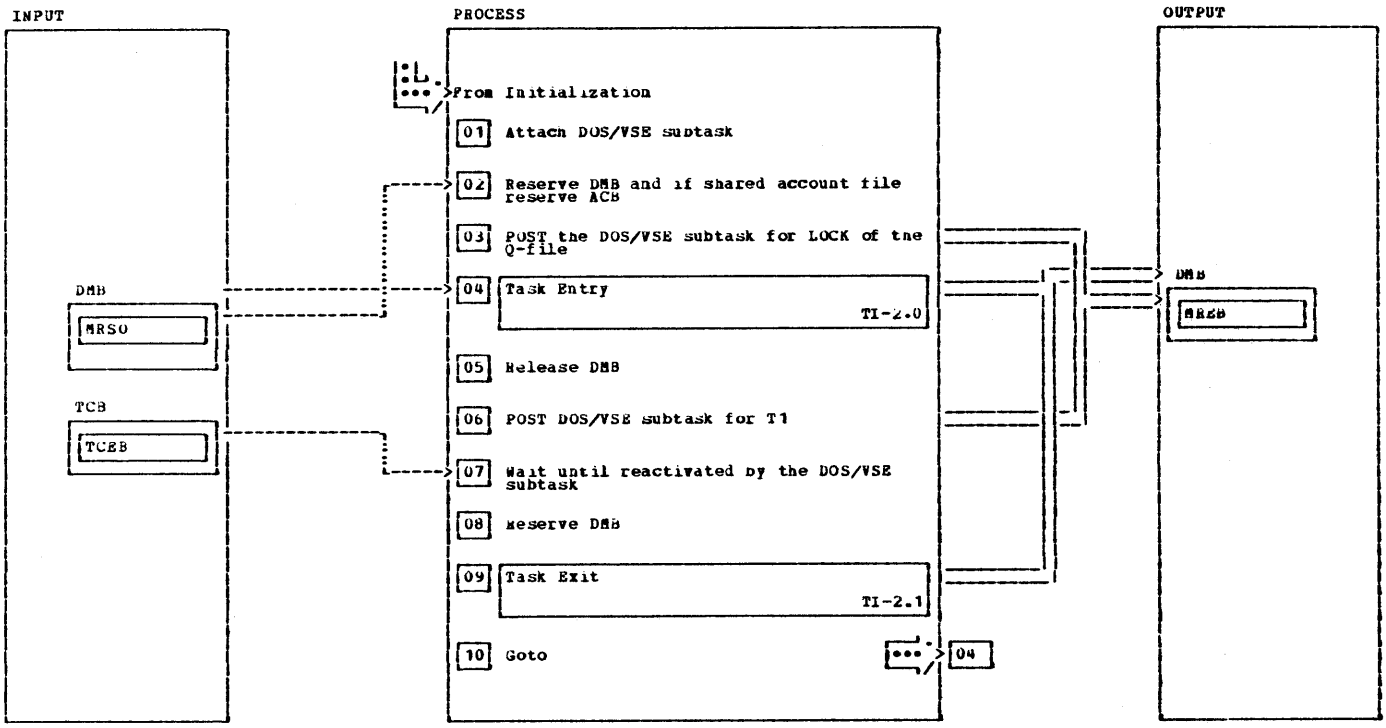


NOTES	MODULE	LABEL	REF
1 Close logical interface		SY90	$CLI
3 Msg.: 1Q29I END OF INPUT ON task,cuu will be issued.			$WTO

NOTES	MODULE	LABEL	REF
4 Msg.: 1Q33I STOPPED task,cuu will be issued by termination routine			$WTO



IPW$$TI - Timer Task Mainline

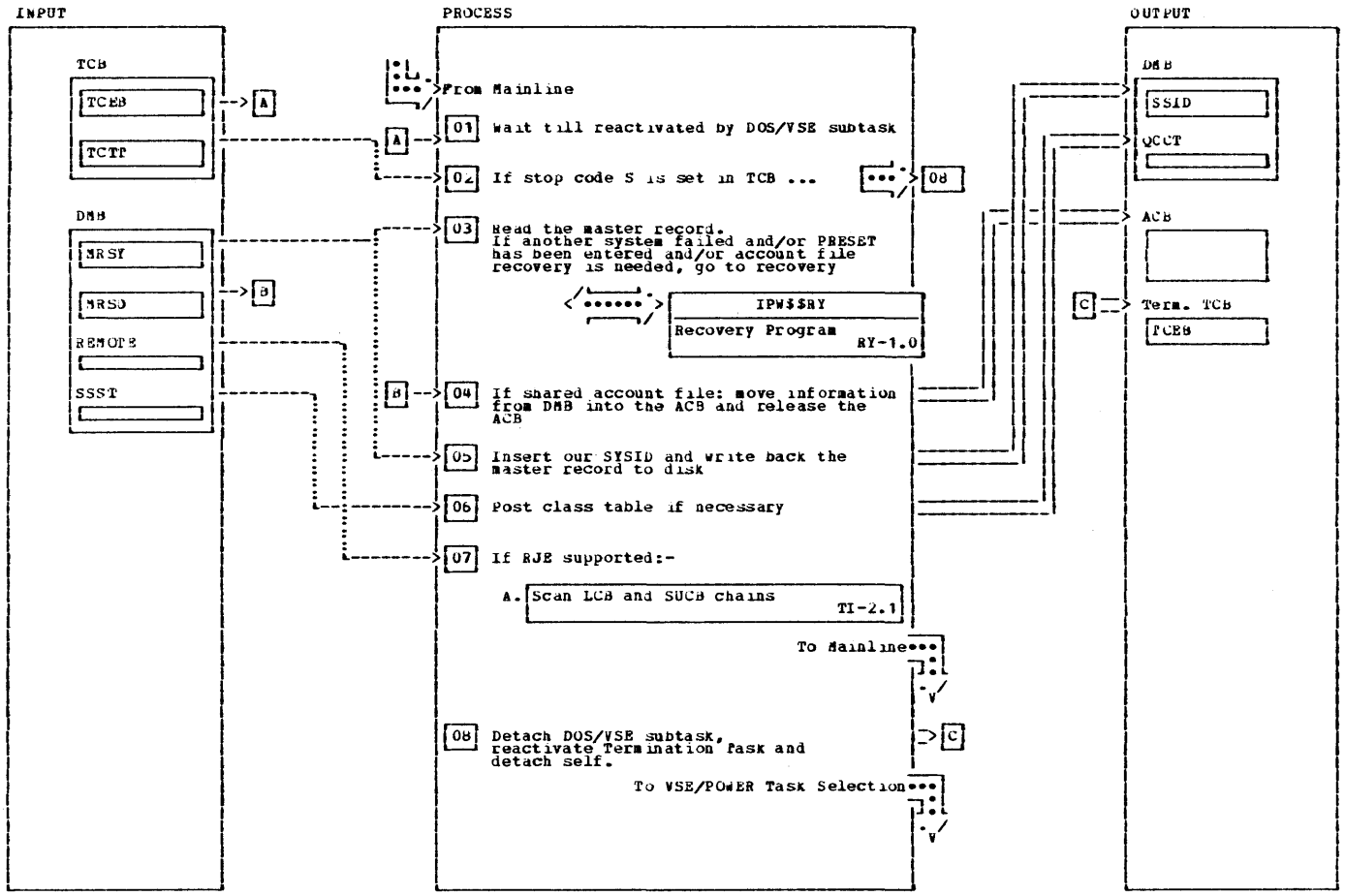


NOTES	MODULE	LABEL	REF
1 The first time the timer task receives control, it must attach a DOS/VSE subtask which will be used to issue the LOCK and SETIME requests so that the VSE/POWER main task does not enter a wait. If no subtask is currently available, VSE/POWER is not able to do its time slicing and is terminated (VSE/POWER cancel macro IPW$CNC) after issuing the message 1QA0I NO SUBTASK AVAILABLE FOR TTTTT. Control is given to the DOS/VSE subtask which later returns control to the timer task.	ATTACH		$GAM $CNC
2 An initial reserve DMB is requested in preparation for the cycle of LOCK,RELEASE - RESERVE,UNLOCK during normal processing. In case of shared account file support, the ACB also must be reserved.		TI10	$BSR
3 The DMB must be acquired for exclusive use by this processor. This is done by issuing a LOCK request for the Q-File. Return from the LOCK is only received when the LOCK is complete or an error has occured. Therefore the subtask must be posted.	POST	TI15	
4 Entry is made at this point when the timer task was in the wait state and 1. either a DMB reserve request was received or 2. after a time interval of T3 seconds.			

NOTES	MODULE	LABEL	REF
5 To prevent any other task in our processor from updating the DMB while it was being used by another processor, we had made a reserve of the DMB. We now own the DMB and so we make a release of the resource so that all tasks waiting on the DMB can process again.		TI60	$RLR
6 The maximum time allowed to one processor to have control of the DMB is governed by time interval T1 (specified at VSE/POWER generation time). The DOS/VSE subtask is activated to post VSE/POWER again after time interval T1 has elapsed.	POST		
7 During the time interval T1 the Q-file is available exclusively for update activities for this processor. If there is no work to do, the interval is shortened.			$$PC
8 When the timer task is again activated, after time T1 seconds or when there is no further work for this processor, a reserve is again made on the DMB to prevent further updating, in preparation for the UNLOCK of the Q-file.			$BSR
9 Exit is made from the routine after another DMB reserve request was made or after time T3 has expired.			

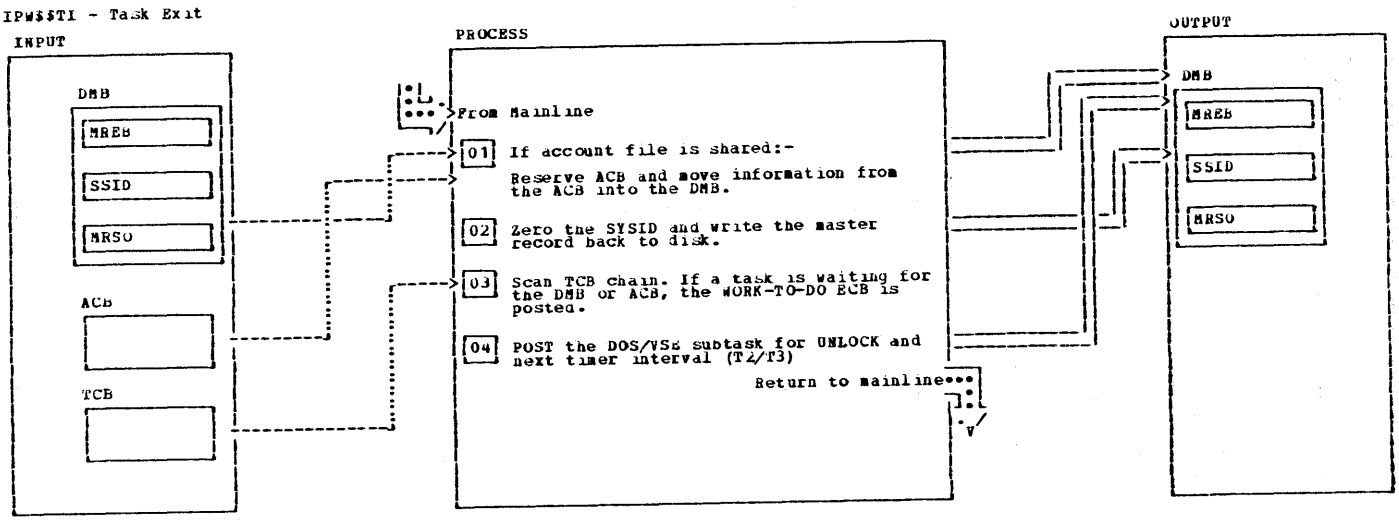
This page was left blank intentionally.

IPW$$TI - Task Entry



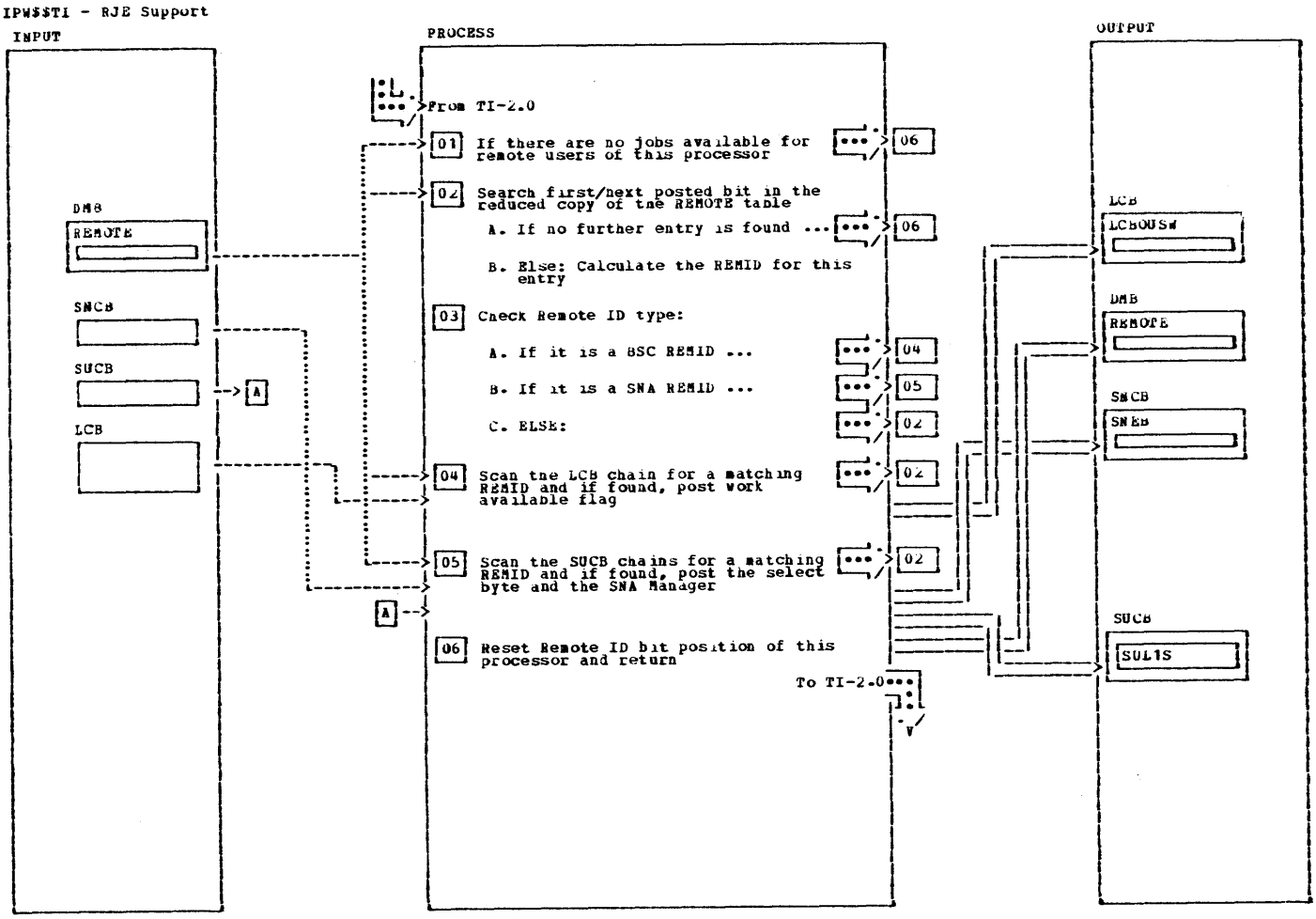
IPW$$TI - Task Entry

NOTES	MODULE	LABEL	REF
1 The DOS/VSE subtask performs the LOCK for the Q-file. Until completion the other VSE/POWER tasks can run if they use the Q-file in read only mode.		TI20	$WFC
2 If PEND has been entered then the timer task must be detached. The stop code S is set by the termination task when all VSE/POWER tasks have completed their work.			
3 Now this processor has exclusive write access to the Q-file. The master record is read from disk and checked for possible abnormal conditions. If the SYSID is not zero then it means that another processor abnormally terminated when it was owner of the Q-file. It may mean that the Q-file chain pointers are not correct. If the PRESET command has been entered the command processor passes one or more SYSIDs via the PRESET SYSID TABLE (label TCW1 of TCB). A nonzero SYSID from the DMB is added to this table and the recovery program is called in order to delete corresponding x-state queue sets and repair possible broken queue-file chains. If the account file is shared and the account-file SYSID field of the DMB is not equal to zero the Recovery Program is entered again; the account file is scanned for the first EOF record in order to update the ACB with valid position and capacity information to continue writing account records without deletion of account of another processor.			
4 If the account file is shared and resident on CKD device, the record address, the residual-record capacity, and the residual-track capacity must be moved from the DMB into the ACB. If the account file is shared and resident on FBA-device, the current block number, the current residual capacity, and the free space in CI are moved from the DMB into the ACB. The ACB is then released.		TI50	$RLR
5 The VSE/POWER SYSID is inserted into the master record in position SSID and, if the account file is shared, also in position SSAC. The master record is written back to disk. If VSE/POWER abnormally terminates, it is possible for another processor to recognize the fact and do recovery.		TI70	$WTQ
6 The live bit of a class-table entry is unposted if there is no work to do for a system which is scanning a class chain. In a shared environment, one class chain can hold work for different target systems. All systems are waiting for class-table posting if relevant tasks have been started. To enable automatic dispatching of such tasks, a bit in the SYSID-CLASS-TABLE is set so that each processor can recognize a work-to-do condition. The timer task scans that portion of the SYSID-CLASS-TABLE which corresponds to the own SYSID and performs posting of the class table according to the bit setting in the SYSID-CLASS-TABLE.			
7 If RJE support is included in the VSE/POWER generation and there is remote output waiting, then VSE/POWER must try to find a REMID that matches the entry in the REMOTE table. If it is found, the output-available flag is set.		RJ00	
8 If the Q-file is still locked, the DOS/VSE DETACH implies an UNLOCK. The termination task is reactivated in order to finish up termination processing.	DETACH	TI200	$DET



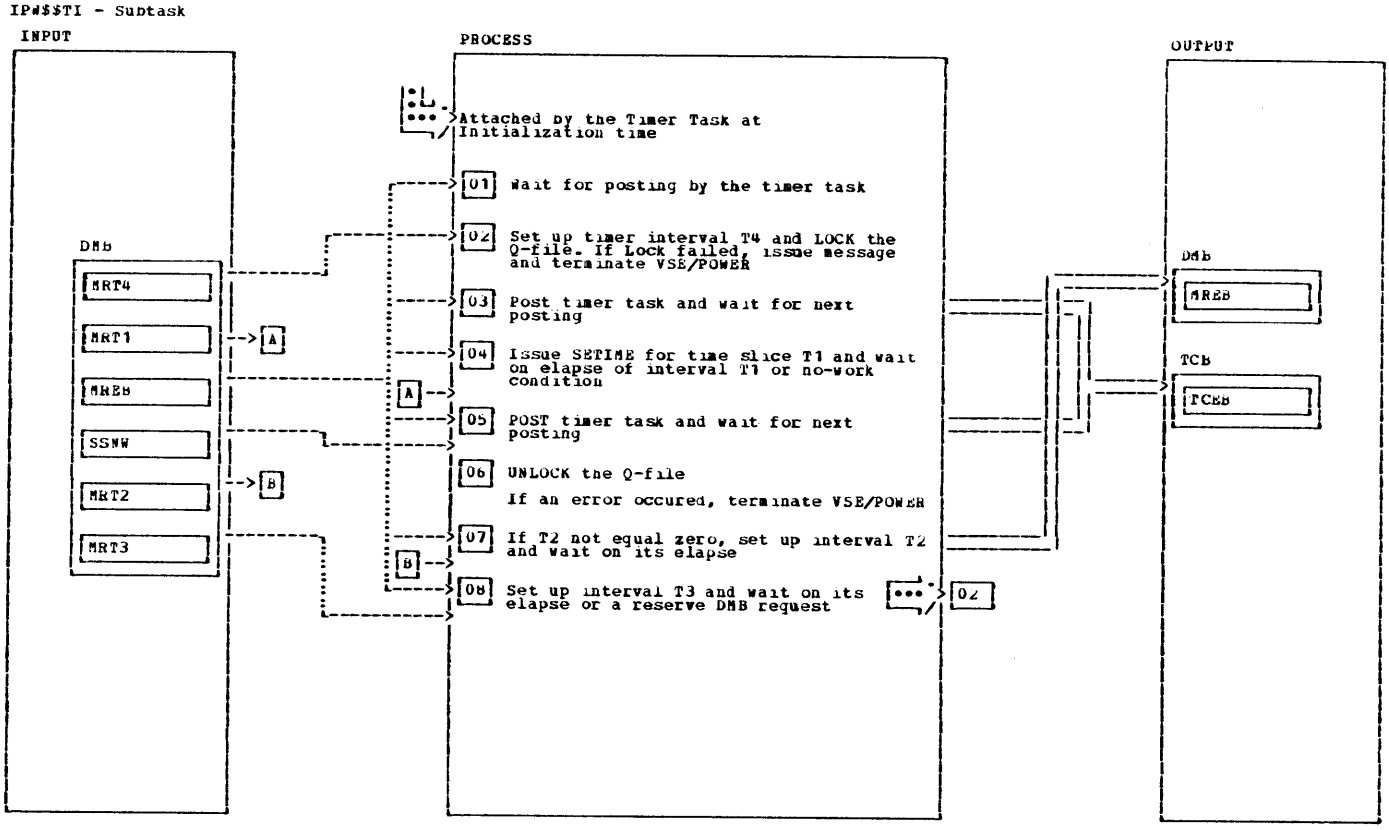
NOTES	MODULE	LABEL	REF
1 If the account file is shared and resident on a CKD-device, the current disk address, the current residual record capacity, and the current residual track capacity are moved from the ACB into the DMB. If the account file is shared and resident on an FBA-device, the current block number, the current residual capacity, and the free-space in CI are moved from the ACB into the DMB.			
2		TI90	$WTQ
3 Posting of the WORK-TO-DO ECB causes skip of waiting for elapse of time interval T3 within the DOS/VSE Subtask.		TI93	

NOTES	MODULE	LABEL	REF
4 The VSE/POWER processor must wait at least T2 seconds before it can issue another Lock. This is to allow a slower processor the chance to get access in competition with a faster processor. T3 is the time after which VSE/POWER must issue a LOCK, if there was no reserve DMB request, to enable it to check that no other processor has put any work in the Queue which must be processed by this processor.	POST	TI100	



NOTES	MODULE	LABEL	REF
1 There may be work for a remote user on the VSE/POWER processor which has been prepared by another processor. In this case the user on the VSE/POWER processor has not been posted and is maybe waiting for work. The REMOTE table within the DMB has bits posted corresponding to all remote IDs of terminals which have work to do in the queue. A copy of this table is reduced by logical and (NC) to a bit setting corresponding to logged on terminals on the VSE/POWER processor. If there is no ID for this processor, VSE/POWER returns to task-entry processing.			

NOTES	MODULE	LABEL	REF
4 If a match on REMID is found then a flag in the LCB is set to indicate that work is available		RJ55	
5		RJ70	
6		RJ96	



NOTES	MODULE	LABEL	REF
1 The first time the DOS/VSE subtask receives control, it has to establish the addressability of CAT,DMB,TCB. It then waits on the first lock request from the timer task.	WAIT		
2	SETIME LOCK STXIT EXIT EXCP WAIT	TS05	
Time interval T4 is needed to get back control from DOS/VSE if a LOCK request takes more time than T4 seconds. In this case there may be an abnormal condition on another processor, (e.g. the Q-file may be locked by another VSE/POWER and this system is unable to do its UNLOCK). The possible reasons are: - Recovery in progress - Save account in progress - Any wait condition - processor-loop in a partition with higher priority A timer-interrupt-driven warning message will be issued.- If the LOCK table is full, message 1Q84I LOCK TABLE SPACE EXHAUSTED is issued and interval T4 is entered again. If another LOCK error occured, message 1QB5I INTERNAL MACRO CALL FAILED RC=RRMM is issued and VSE/POWER is cancelled. However, before setting up T4, it is checked if PEND has been entered. Then the stop code 'S' is found in the TCB and the normal processing is left via posting the Timer Task for termination.	CANCEL		
3	POST	TS30	

NOTES	MODULE	LABEL	REF
The timer task gains control at task entry step.	WAIT		
4 Control comes from posting for interval T1. The DOS/VSE subtask waits on elapse of the time slice T1 or the no-work condition. The no-work condition ECB is posted by the nucleus just before the SVC 7 is issued. During interval T1 VSE/POWER is dispatchable and can do its work with exclusive write access to the Q-file	SETIME WAITM		
5 The timer task gains control at reserve DMB request to allow another processor to get write access to the Q-file.	POST WAIT	TS40	
6 The timer task has done all necessary updates to the DMB and VSE/POWER can unlock the Q-file to allow access by the other CPUs. If an error occured issue message 1QB5I INTERNAL MACRO CALL FAILED RC=RRMM .	UNLOCK CANCEL		
7 If T2 not equal zero,VSE/POWER waits on elapse of T2 seconds. However, before setting up T2 and after elapse of T2, it is checked if PEND has been entered. Then the stop code 'S' is found in the TCB and the normal processing is left via posting the Timer Task for termination.	SETIME WAIT	TS50	
8 If there is no work to do, VSE/POWER waits on elapse of T3 seconds or a DMB reserve request. If the nucleus recognizes a reserve DMB request, it posts the work to do ECB.	SETIME WAITM	TS60	

IPW$$TR - VSE/POWER Task Terminator	
Label	Routine
TRCS	Entry processing and loop check
TR08	Register house-keeping
TR17	Check file type in which I/O error occurred
TR26	Account file error recovery
TR38	Queue file error recovery
TX00	Data file error recovery
TX20	Queue/Data file clean up
TX36	Write Account record
TX78	Execution writer wrap up
TY00	Account processing termination
TY20	Close and unassign of tape device
TY30	Unassign unit record device
TY38	Release all resources
TY12	Execution Reader wrap up
TY48	PNET receiver / transmitter wrap up
	Release all workspaces (real - virtual)
	Final wrap up

Services Used	
Service	Macro
Task Management	IPW$DET IPw$WFC
Resource Management	IPW$RLR IPw$RSR
Storage Management	IPW$RLW IPW$RSW IPW$RLV
Message Service	IPw$GAM IPW$WTO
Disk / Tape Service	IPW$WTQ IPw$RDQ IPW$CTT
Timer Service	IPw$RDC

Functions used	
Module	Macro
IPW$$AQ	IPW$AQS
IPW$$AT	IPW$CNC
IPw$$AS	IPW$IAS
IPW$$DQ	IPW$DQS
IPW$$FQ	IPw$FQS
IPW$$GA/GF	IPW$CAF
IPW$$LU	IPW$ULP
IPW$$PA/PF	IPW$PAR
IPW$$PD	IPw$PDR

Called by	
Module	Description
IPW$$NU	VSE/POWER Nucleus when an I/O error occurred
IPW$$LW	Logical Writer
IPW$$LR	Logical Reader
IPW$$ER	Diskette Reader

Labels	Chart TR: IPW$$TR - Task Terminator		Modified Data Fields	Reg. Usage	Calls
	Entry to this routine is from any failing VSE/POWER routine.				
TRSD	The first 16 bytes constitute the section descriptor:				
	'TRCS release'				
	Register Usage:				
	0: work register			R0	
	1: work register			R1	
	2: work register			R2	
	3: work register			R3	
	4: Module control block pointer			R4	
	5: Queue record pointer			R5	
	6: Disk management block pointer			R6	
	6: partition control block pointer				
	7: work register			R7	
	8: work register; link register			R8	
	9: base register			R9	
	10: pointer tp permanent area			R10	
	11: address of TCB			R11	
	12: reserved for Nucleus use			R12	
	13: address of save area			R13	
	14: link register			R14	
	15: 2nd base register			R15	
TR04	If the failing task is the initiator task or the command processor task, recovery is not possible, message 1Q82I I/O DURING ttt, VSE/POWER TERMINATED is printed and the job is canceled..... >	TR07		R3	IPW$GAM
TR06	If this routine has been entered before (meaning that more than one unrecoverable I/C error has occurred), message 1Q75I MULTIPLE TERMINATION OF TASK, VSE/POWER TERMINATED is printed and the job is canceled..... >	TR07		R3	IPW$GAM
	Otherwise branch to.....>	TR08			
TR07	Cancel VSE/POWER				IPW$CNC

Labels	Chart TR: IPW$$TR - Task Terminator		Modified Data Fields	Reg. Usage	Calls
TR08	The addressability of the save area is set up in register 2 and the failing task's registers 0 and 1 in SVR0 and SVR1.		SVR0(IPW$DSV) SVR1(IPW$DSV)	R2	
	If the TCB is not owned by IPW$$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 = PUN), 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', C'I', or C'H') 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$SDV)		
TR16	Now space is reserved for a new register save area, addressed by register 13, and the TR workarea.			R13	IPW$RSW
	The TR workarea is used to save: • Some TCB fields • Disk address of bad Q-record • Message formatting area The TR workarea is anchored to the TCB.		TCMW(IPW$DTC)		
	If it is a Save account task and termination code is 'C', branch.....>	TR27			

Labels	Chart TR: IPW$$TR - Task Terminator		Modified Data Fields	Reg. Usage	Calls
TR17	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	
	Otherwise, the VSE/POWER cancel code is set to 'canceled due to unrecoverable I/O error' (QRCN = X'70').		QRCN(IPW$DQR)		
	Addressability of the synchronous save area is set up in register 1, register 2 is set up as MCB counter (set to 8), and register 3 is pointed to the first MCB (A(CAQ1)).			R1,R2 R3	
TR18	Now a check is made to see if the MCB of the failing task indicates a system MCB. If so, branch to..... >	TR22			
	If the failing function is in the account file, branch to..... >	TR26			
	Otherwise (failing device is not a system module), branch to ..... >	TX22			
TR22	The 'failing device found' switch (X'FF') is set in the work space (address in register 1) and a branch table is used to branch to the proper routine. The failure may have occurred in:		TRFD	R1	
	• Queue file; branch to.....>	TR38			
	• Any data file; branch to.....>	TX00			
TR26	Account File Routine				
	Message 1Q61I IRRECOVERABLE ERROR ON AFILE N CUU is printed in the.....> subroutine.	TY70		R14	
TR28	If no task is waiting for an account function, branch to..... >	TR31			
	If a task was waiting for a put account function, branch to ..... >	TR30			
	If a task was waiting for a get account function, the termination type in the TCB is set to 'immediate stop' (TNTT = C'S').		TNTT(IPW$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			
TR31	If there is no queue space for save account, (TCTT='C'), branch.....>	TX22			

Labels	Chart TR: 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 issued.....>	TY70	MRJA(IPW$DQC) CAPA(IPw$DPA)		IPw$GAM
	If the account function was not save account, branch to.....>	TX22			
	Otherwise, If the account file was saved into the Punch queue, branch..>	TY00			
	In all other cases, branch.....>	TX22			
TR38	Queue Record Routine If Q-file resides on an FBA device, get disk request word out of MCB. The CCW real address is obtained from the CCB addressed by register 1 and placed in register 1. Using the VIRTAD macro, the corresponding virtual address is obtained from the real address. Then the address of the complete MBBCCHH field is obtained in register 8. If this address is not that of the master queue record, branch to..... >	TR42		R1 R8	
	Otherwise, the error occurred in the master queue record and VSE/POWER must be canceled. Therefore, the DMB is reserved and message 1Q63I IRRECOVERABLE I/O ERROR IN QUEUE MASTER REC - CUU is printed.....>	TY70		R14	IPw$RSR
TR40	Message 1Q76I VSE/POWER CANNOT CONTINUE is printed and VSE/POWER is abnormally terminated.				IPW$GAM IPw$CNC
TR42	Message 1Q61I IRRECOVERABLE I/O ERROR ON QFILE N CUU is printed in the... >	TY70		R14	
	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 during a 'get next queue record' function (TCFT=C'N'), branch to..... >	TR56			
	If the error occurred during 'get next queue record in set' (TCFT=C'S') branch to..... >	TR50			
	If the error did not occur during a 'free queue set' function, branch to..... >	TR66			

Labels	Chart TR: IPW$$TR - Task Terminator		Modified Data Fields	Reg. Usage	Calls
	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 and the partition is canceled via a branch to	TR40			IPW$GAM
TR50	Delete Specific Queue Set from System Files				
	The disk management block (DMB) is locked.				IPW$RSR
	The previous Q-record pointer and class-id are saved.		QCQW(IPW$DMC) QCCL(IPW$DMC)		
	When the failing queue record is not the first-in-set, the first-in-set queue record is read in.				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$DTC)		
	The DMB is released.				IPW$RLR
	Message 1Q64I JOB jobname RDR/PUN/LST SET DELETED is printed in the.....	TY70		R14	
	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 saved			R3	
	in TCGW. Then queue space is reserved if necessary and the address of the		TCGW(IPW$DTC)		IPW$RSW
	task class list is loaded in reg 4.			R4	

Labels	Chart TR: 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 ..... >	TX26		R7	
TR60	When the queue set in error is found, it is deleted in the ..... >	TR76		R8	
	subroutine and message 1Q65I UNKNOWN RDR/LST/PUN SET DELETED is printed in the..... >	TY70		R14	
	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$DTC)	R8 R7 R8	IPW$RSW
	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		R14	
	subroutine. Then the DMB is released and, if the current (not the failing) record is the first-in-set, branch to..... >	TR72			IPW$RLR
	Otherwise, if the failing record was the first-in-set, no more queue file clean up is required, so branch to..... >	TX36			
TR72	If the function trace indicator shows that an add to queue function was being processed at the time of the failure (TCFT = C'A'), the request is executed and branch is to..... >	TX36			IPW$AQS
TR74	Otherwise, the queue set is deleted from the queue and a branch is made to..... >	TX36			IPW$DQS

Labels	Chart TR: IPW$$TR - Task Terminator		Modified Data Fields	Reg. Usage	Calls
	Delete Bad Record in Chain:				
	Registers used by this routine are:				
	1: relative record address			R1	
	2: address of field containing absolute disk address			R2	
	5: address or 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 record is found, branch to..... >	TR82			IPW$RDQ
	If the end of the chain is found before a bad record is hit, the task is detached at..... >	TR96			
TR80	Set the queue record seek address to zero.		QCQW(IPW$DQC)		
TR82	Set up queue space addressability in register 5. Get the relative last-in-queue pointer in register 1			R5	
	and go to..... >	TX18		R1	
	to have the address converted (absolute address in TCQW). If the bad record is not the last record, branch		TCQW(IPW$LTC)		
	to..... >	TR84			
	Zero the last, and bad, record. If the queue set constitutes the entire class, the entire class is deleted		TCQW(IPW$DTC)		
	and a branch is made to..... >	TR94			
TR84	Otherwise, read a new record and, if the previous record was the bad one,				IPW$RDQ
	branch to..... >	TR88			
	If the beginning of the chain is reached (reading was backward) before a bad record is found, the task is				
	detached at..... >	TR96			
TR88	The reverse pointer (to the bad record) is updated to point to the queue set before the bad record and		QRQP(IPW$LQC)		
	then written back. If the bad record was the first in chain, the new address is converted from absolute to				IPW$WTQ
	relative in routine..... >	TX16			
	and updated in the chain.				
TR90	The next in queue pointer is updated and the record written back.		QCQN(IPW$DQC)		IPW$WTQ
	Branch to..... >	TR94			

Labels	Chart TR: 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
	and the last-in-class pointer is set, as well as the active indicator.				
TR94	Indicate record found and pass control back to the calling routine... >	R8		R1	
TR96	Indicate Q-record not found			R1	
	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
	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 (RDR, LST, PUN or XMT) class table is loaded into register 7.			R7	

Labels	Chart TR: IPW$$TR - Task Terminator		Modified Data Fields	Reg. Usage	Calls
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$DTC)	R8	
TX10	If the entry is just the first in the chain, the absolute address of the next in chain is obtained in register 2 and converted to relative in..... >	TX16		R2	
	This relative address is returned in register 1 and saved as the new first in set address, the active indicator is set (CTQL = X'80'), and branch to..... >	TX14	CTQF(IPW$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$RDQ IPW$WTQ
	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$RDQ IPW$WTQ
	Disk Address Conversion Routine (absolute to relative)				
	Registers used by this routine are:				
	0: work register			R0	
	1: relative record class			R1	
	2: absolute record address pointer			R2	
	3: module control block address			R3	
	14: return register.			R14	

Labels	Chart TR: IPW$$TR - Task Terminator		Modified Data Fields	Reg. Usage	Calls
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	
	Disk Address Conversion Routine (relative to absolute)				
	Registers used by this routine are:				
	0: work register 1: relative record address 2: pointer to field containing absolute address 3: module control block address 14: return register.			R0 R1 R2 R3 R14	
TX18	The high order bits are stripped of the absolute address, the DMB address is set in register 3. Then the relative track and relative cylinder number are calculated in register 0; 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$DQC)	R3 R0 R2 R14	
TX20	Clean the Queues				
	This routine is entered at TX22 in the case of unrecoverable I/O error. This entry point (TX20) is for PSTOP. Therefore, 'canceled due to PSTOP' is set in the VSE/POWER cancel code = (QRCN = X'30').		QRCN(IPW$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			

Labels	Chart TR: IPW$$TR - Task Terminator		Modified Data Fields	Reg. Usage	Calls
	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			
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..... >	TX26		R1	
	If this 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	
	Branch to..... >	TX36			
TX26	General Exit from Queue/Data File Clean Up				
	If the task was an execution processor, branch to ..... >	TX78			
TX32	If RJE task, branch to..... >	TY38			
	If task was not a RDR, LST, or PUN task, branch to..... >	TX34			
	If tape spooling or other tape function, branch to..... >	TY20			
	Otherwise..... >	TY30			
	Otherwise, if the device used by the task is a tape , 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 the status task (TCTI='P PS')..... >	TY30			
	If not, the task is an unknown task; in that case branch to..... >	TY38			

Labels	Chart TR: IPW$$TR - Task Terminator		Modified Data Fields	Reg. Usage	Calls
TX36	Write Account Record:				
	If the task was owned by an execution processor, the accounting function is ignored; branch to..... >	TX26			
	If the task is a receiver/transmitter or Poffload task, branch to.....>	TX32			
	If this is a print status task.....>	TX26			
	The current time value is obtained via an IPW$RDC call, and then stored in the queue record.		QRET(IPW$DQR)		IPW$RDC
	The virtual queue file address is loaded into register 1, and the length of the account record which depends on the class type (RDR,LST, or PUN) is determined and placed in register 0. If it is none of these class types, branch to..... >	TX34		R1	
	In case of stopped list or punch task, counters for pages, lines/cards, and number-of-copies are updated and moved into the queue record.		QRNE(IPW$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..... >	TX39D			
	The account record that could not be written before is now written.				IPW$PAR
TX39D	If task is not a RDR task, but delete queue set must be done, it is performed now. branch to ..... >	TX32			IPW$DQS IPW$FQS
TX40	Input Mode Data/Queue File:				
	If TCTI = C'P PS', no clean up is necessary, so branch to..... >	TY26			
	Otherwise, if the function is neither punch nor list, branch to..... >	TX52			
	If the device used is tape, branch to ..... >	TX36			
	If a PSTOP command without RESTART was issued, branch to..... >	TX51			
	Otherwise (TCTT = C'R'), get the address of the logical list save area in register 1 and point register 4 to the account counter DSECT (LADS).			R1,R4	
	Read the first queue record.				IPW$RDQ
	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	
	If restart is not active, update the current page or card, depending on the class type.		LACP LACR	R0	

Labels	Chart TR: 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 deleting is not required, branch to..... >	TX54			IPW$DQS
	Otherwise, after deleting, branch to..... >	TX56			
TX54	If at this stage a 'release queue set' function is not indicated (TCFT = C'F'), branch to ..... >	TX58			
	The already deleted queue set is freed and a branch is made to..... >	TX36			
TX58	A test is made to see if the queue set has been released successfully. To do this, the DMB is once again reserved, the first in set queue record is read and the queue record identifier is inspected to see if the queue set was freed (QCQI = C'F') or reused for another queue set. Then the DMB is released again and, if the queue set was not freed before, it is freed now.				IPW$RSR IPW$RDQ IPW$RLR IPW$FQS
	At the successful completion of this routine, branch to..... >	TX36			

Labels	Chart TR: IPW$$TR - Task Terminator		Modified Data Fields	Reg. Usage	Calls
TX68	Routine for Non-Tape Device and Non-Deleted Queue Set				
	The DMB is reserved. The first record in the queue set is now read.				IPW$RSR IPW$RDQ
	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. The restart information and the current copy group index are saved in the first-in-set queue record. If this task does not belong to an execution reader, branch to..... >	TX76	QRXS(IPW$DQR) QRCN(IPW$DQR) QRNC(IPW$DQR) QRRR(IPW$DQR) QRCI(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$DQR)		
TX76	The queue record is written.				IPW$WTQ
	If RJE task, then branch to..... >	TX64			
	If reader queue record then branch to ..... >	TX64			
	If list or punch queue record then the line bit in the master class table is set, the DMB is released and a branch is made to..... >	TX36	CTQL(IPW$DQC)	R3	
	Execution Writer Wrap-up:				
TX78	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..... >	TY38		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			

Labels	Chart TR: IPW$$TR - Task Terminator		Modified Data Fields	Reg. Usage	Calls
TX80	If no output was generated, release the queue set, and branch to..... >	TX86			IPW$FQS
	Otherwise (bad record somewhere in the queue set), segmentation of the queue set is effected. This is done by forcing end-of-data (blank data field, length of one, NOP CCW) and setting an 'I' disposition to 'D', while at the same time setting the queue record identifier to 'reader'.		TCCC(IPW$DTC) TCGP(IPW$DTC) QRDP(IPW$DQR) QRQI(IPW$DQR)		IPW$PDR
	Now the queue set is added to the class chain; branch to..... >	TX86			IPW$AQS
TX84	Message 1Q61I IRRECOVERABLE I/O ERROR ON CUU is printed in the..... > subroutine.	TY70			
TX86	Control of the device in the user partition being spooled is passed back to the execution reader by setting the address of the controlling TCB (that is in the device list entry IPW$DDE) in the partition control block (IPW$DPD) to point to the execution reader TCB. The execution reader task will initiate an execution writer task to continue processing. The execution reader is set dispatchable, as well as the partition. Then messages 1Q68I SEGMENTATION FORCED FOR jobname nnnnn ttt CUU and 1Q69I DEFAULT OPTIONS TAKEN FOR jobname nnnnn ttt CUU are printed.		TLTC(IPW$DTL) TCEB(IPW$DTC)		IPW$GAM IPW$GAM
	If the device is tape , unassign the device using registers 0, 1, and 3. Then branch to..... >	TY38		R3,R1	IPW$ULP
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.... >	TY62			
	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

Labels	Chart TR: IPW$$TR - Task Terminator		Modified Data Fields	Reg. Usage	Calls
	Message 1Q66I ACCOUNT FILE KEPT and message 1Q72I PACCOUNT TERMINATED are printed.				IPW$GAM IPW$GAM
	Reset save account active indicator If the activity was on disk, or if output spooling was active, branch to..... >	TY38	ACTB(IPW$DAC)		
	Otherwise, the CCB pointer is obtained in register 3; if no CCB is initialized, branch to..... >	TY08		R3	
	If the activity was not on tape, branch to..... >	TY06			
	Otherwise, the CCB pointer for the tape is stored in register 3 and set to a dummy SVC value. Then a fetch is simulated to close the SA tape file. This is done via $$BCLOSE.			R3	$$BCLOSE
	First storage for a service request is acquired and formatted. the SRB then passed to asynchronous service for processing. On return the SRB is released.			R14,R15	IPW$RSW IPW$IAS IPW$RLW
TY06	Unassign the device and branch to..... >	TY38			IPW$ULP
TY08	Set up a dummy CCB and point register 3 to it; then unassign the device and branch to..... >	TY06		R3	IPW$ULP
TY20	Close and unassign tape :				
	Register 7 is based on the tape control block. If no unrecoverable I/O error was encountered (IPW$$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			
TY22	The command is issued to rewind and unload the tape.				IPW$CTT
TY24	The tape unit is unassigned; then branch to..... >	TY30			IPW$ULP
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. Additionally, if a ECB was supplied to be posted on completion of the print status task, the ECB is posted				IPW$GAM
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.. >	TY36D			

Labels	Chart TR: IPW$$TR - Task Terminator		Modified Data Fields	Reg. Usage	Calls
	Unassign the Unit Record LUB:				
TY30	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..... > subroutine.	TY70			
TY32	Message 1Q71I TTT,CUU TERMINATED is printed. branch to.....>	TY36			IPW$GAM
TY34	Message 1Q33I STOPPED TTT,CUU is printed.				IPW$GAM
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			
	If the task was the SYSIN or off-loading task, branch to..... >	TY38		R2,R8	
	Load register 8 from register save area.				
	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 task is writer task, branch.....>	TY36C			
	If this buffer is active, then let register 1 point to the other CCB.			R1	
TY36B	If task is not a writer task.....>	TY36D			
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

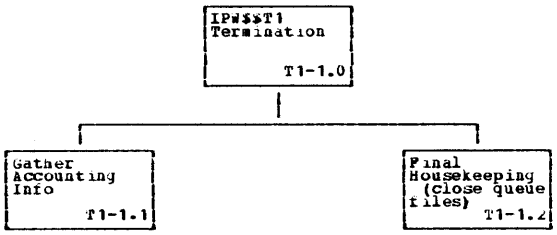
Labels	Chart TR: IPW$$TR - Task Terminator		Modified Data Fields	Reg. Usage	Calls
TY36D	Assignments made for this task are now to be unassigned.				
	Load register 2 with the cuu field from the TCB.			R2	
	Load the branch index to request for a generic unassign into register 0.			R0	
	Load the address of the VSE/POWER PIB into register 1.			R1	
	The current unit record device is unassigned.				IPW$ULP
	If the task is not a reader task, branch to..... >	TY38			
	If the 3540 PWS is not already released branch to..... >	TY37			
	If the reader task has not been activated in combination with a 3540 diskette reader, branch to..... >	TY38			
	Load the address of the register 7 field in the physical save area of the TCB, which can contain the programmer unit number, as a CCB like field into register 3.			R3	
	Point register 1 to the PIB of the VSE/POWER partition.			R1	
	Load register 0 with the branch index to request to unassign the programmer logical unit in the CCB like field addressed by register 3.			R0	
	The programmer logical unit assigned to the 3540 diskette device is unassigned and branch is made..... >	TY38			IPW$ULP
TY37	Otherwise address the 3540 PWS and reload register 3 with the address of PELU - 6, to use this part of the physical work space as a CCB like field.		R8	R3	
	Point register 1 to the PIB of the VSE/POWER 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	
	The programmer logical unit assigned to the 3540 diskette device is unassigned.				IPW$ULP
	If there exists a high level PWS, this 3540 diskette device has to be unassigned too..... >	TY37			
	Otherwise branch to continue..... >	TY38			

Labels	Chart TR: IPW$$TR - Task Terminator		Modified Data Fields	Reg. Usage	Calls
TY38	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.				⊥PW$RLR
				R7,R8	
	A scan through all lockwords is made. If the task owns a lockword, it is released. If the task is an execution writer, scan all data file MCBs and wait for completion of any outstanding double-buffer requests.				
	Execution Reader Wrap-up:				
TY10	If the task is not an execution reader, branch to..... >	TY43			
	Establish addressability of the PCB in register 6.			R6	
	Release DMB to prevent execution writer waiting on locked resource.				⊥PW$RLR
	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(⊥PW$LTL) TNTT(⊥PW$DTC) TNEB(⊥PW$DTC)		⊥PW$WFC
TY14	The VSE/POWER control flags in the user 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			⊥PW$ULP
TY16	Unassign the device specified in the entry.		TCGW(⊥PW$DTC)		⊥PW$ULP

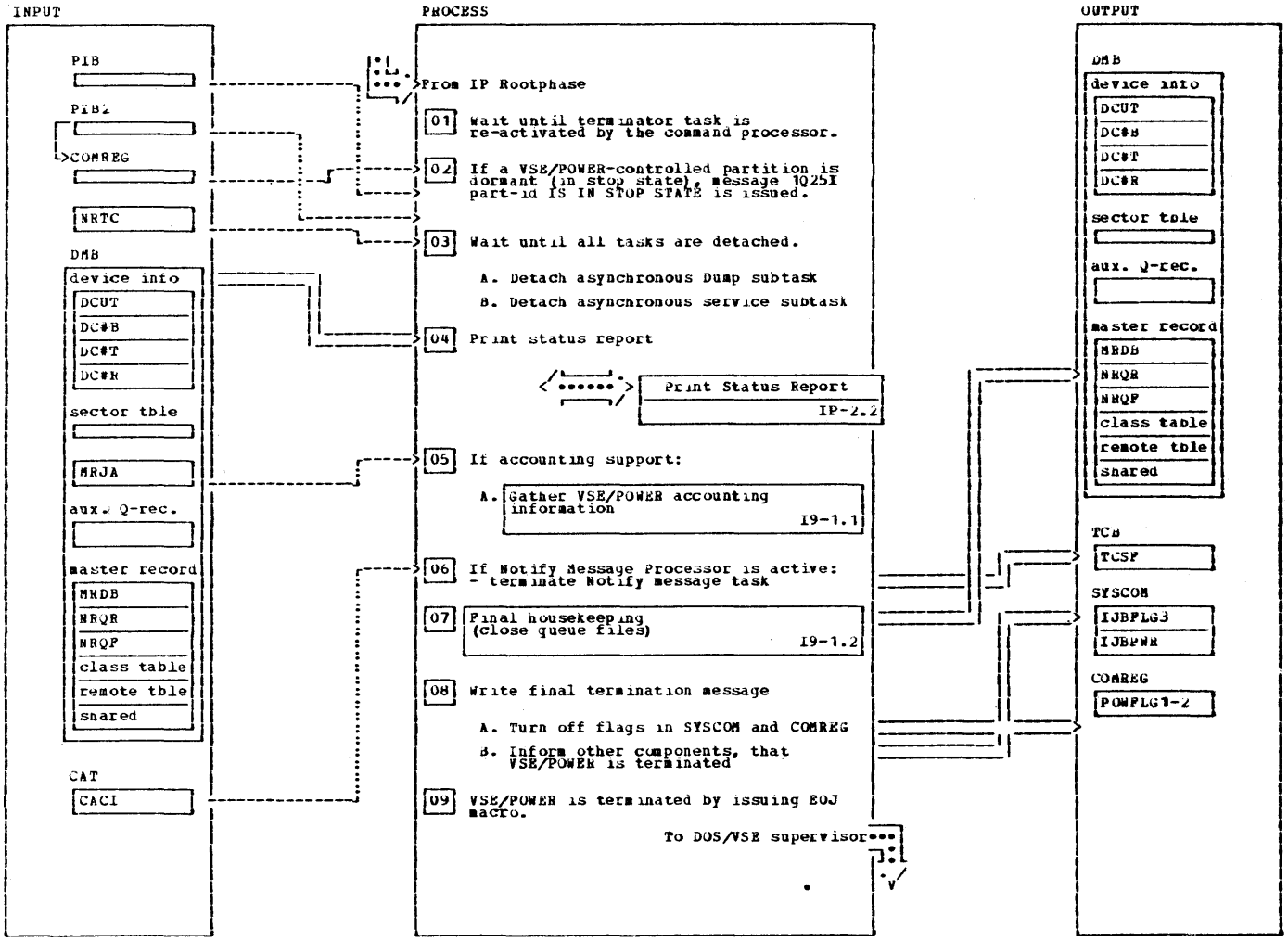
Labels	Chart TR: IPW$$TR - Task Terminator		Modified Data Fields	Reg. Usage	Calls
TY18	Point register 4 to the next entry in the PDB and continue to unassign, via branches to TY16, until all units have been handled.			R4	
	Message 1Q70I TASK FAILURE, STOPPED TTT is printed.				IPW$GAM
	The user partition is canceled				TREADY
	All Data file MCB's are scanned to check whether the execution writer task has any I/O still pending while the task terminated. This is done by locking the appropriate MCB and checking if the task did issue the last I/O. If yes, the execution writer task is put in wait state until the I/O completes, if not already done. If an I/O error occurred, message 1Q61I IRRECOVERABLE I/O ERROR ON cuu is issued by routine...>	TY70		R14	IPW$RSR IPW$WFC
	An Indication is set that the I/O request is handled and the buffer space is released. Finally the MCB is unlocked.		MCTV(IPW$DMC) MC$T(IPW$DMC)		IPW$RLW IPW$RLR
	PNET Receiver/Transmitter wrap up:				
	If task is not a receiver or transmitter.....>	TYPN30			
	Pick up proper address of error exit according to task type. Restore registers and branch to error exit.....>	R14		R14	

Labels	Chart TR: IPW$$TR - Task Terminator		Modified Data Fields	Reg. Usage	Calls
	Release Work Space				
	The address of the first page is loaded into register 8.			R8	
TY44	The address of the 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 is positive (buffer not in use), branch to..... >	TY54			
	Otherwise, load the positive buffer length into register 3.			R3	
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.				IPW$RLW
	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	
	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.				

Labels	Chart TR: IPW$$TR - Task Terminator		Modified Data Fields	Reg. Usage	Calls
TY57	Release all acquired virtual storage				IPW$RLV
	If this is not an RJE task, branch to.....>	TY67			
	If this is not an SNA reader or writer, branch to.....>	TY66			
TY58	If this is an SNA inbound processor task, the address of the inbound processor is loaded into register 2. Otherwise the address of the SNA outbound processor task is loaded into register 2.			R2	
TY64	Addressability of the SNA work area (WACB) is set up in register 3.			R3	
	The address of the RJE,SNA error routine (WAER) is loaded into register 1. The processing switch in the SNA work area is set to X'FE' to indicate that the logical interface is closed. The RJE,SNA error routine is branched to via register 1.		WASW(IPW$DWA)	R1	
TY66	The addresses of the related RJE,BSC routine are loaded from the task save area and a branch is made via register 14.			R1 R14-R9 R14	Phase IPW$$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	
TY64	Addressability of the SNA work area (WACB) is set up in register 3. The address of 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. The RJE,SNA error routine is branched to via register 1.		WASW(IPW$DWA)	R3 R1	
TY68	The task is detached.				IPW$DET
	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

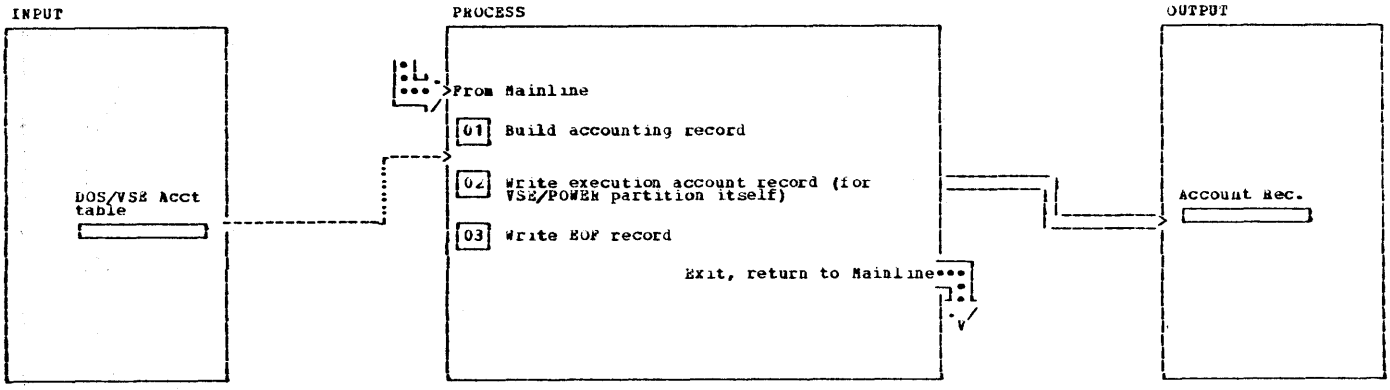


IPW$$T1 - Termination Processing



	NOTES	MODULE	LABEL	REF
1	The command processor dispatches the Initiator/ Terminator task when a PEND command has been entered by the central operator.			$WPI
2	The DOS/VSE PIB table is scanned to see if a VSE/POWER-controlled partition is in the stop state. If s an information message is sent to the operator, who must start again the partition to complete processing of the current queue entry, if any. Message 1Q25I part-id IS IN STOP STATE	COMREG	T100	$WTO
3	This is the case when only two or three VSE/POWER tasks are still active (termination task, command processor task and timer task when shared spooling) The task-detach routine, located in the VSE/POWER nucleus, posts the terminator task only when 2 or 3 tasks are still alive.			$WPC
3A	A internal macro call: IPW$GTE LENGTH=(R0) where R0 is set to zero, is issued to indicate the partly filled Area is to be dumped if required.			$GTE
3A	A internal macro call: IPW$IAS TYPE=DETACH,TASK=DUMP is issued to detach the asynchronous dump subtask, which has been attached at VSE/POWER initialization time.			$IAS
3B	An internal macro call IPW$IAS TPYE=DETACH is issued to detach the asynchronous service subtask, which has been attached at VSE/POWER initialization time.			$IAS
5	VSE/POWER accounting information is gathered from the DOS/VSE accounting table and a final execution accounting record followed by an EOF record is written.			
6	If Notify message task is active, the TCB is released and the Notify task is set dispatchable			
8	Message: 1Q21I VSE/POWER HAS BEEN TERMINATED		T190	$WTO
8A	Reset address of VSE/POWER partition in SYSCOM		T195	
8B	A SUBSID macro instruction is issued, to inform other compenents (e.g ICCF) that VSE/POWER is terminated If a bad return code (¬=0) is given back, message 1Q85I INTERNAL MACRO CALL FAILED, RC=rrmm is issued and VSE/POWER termination is canceled.	SUBSID		
9	VSE/POWER is terminated by issuing an EOJ macro instruction	EOJ		

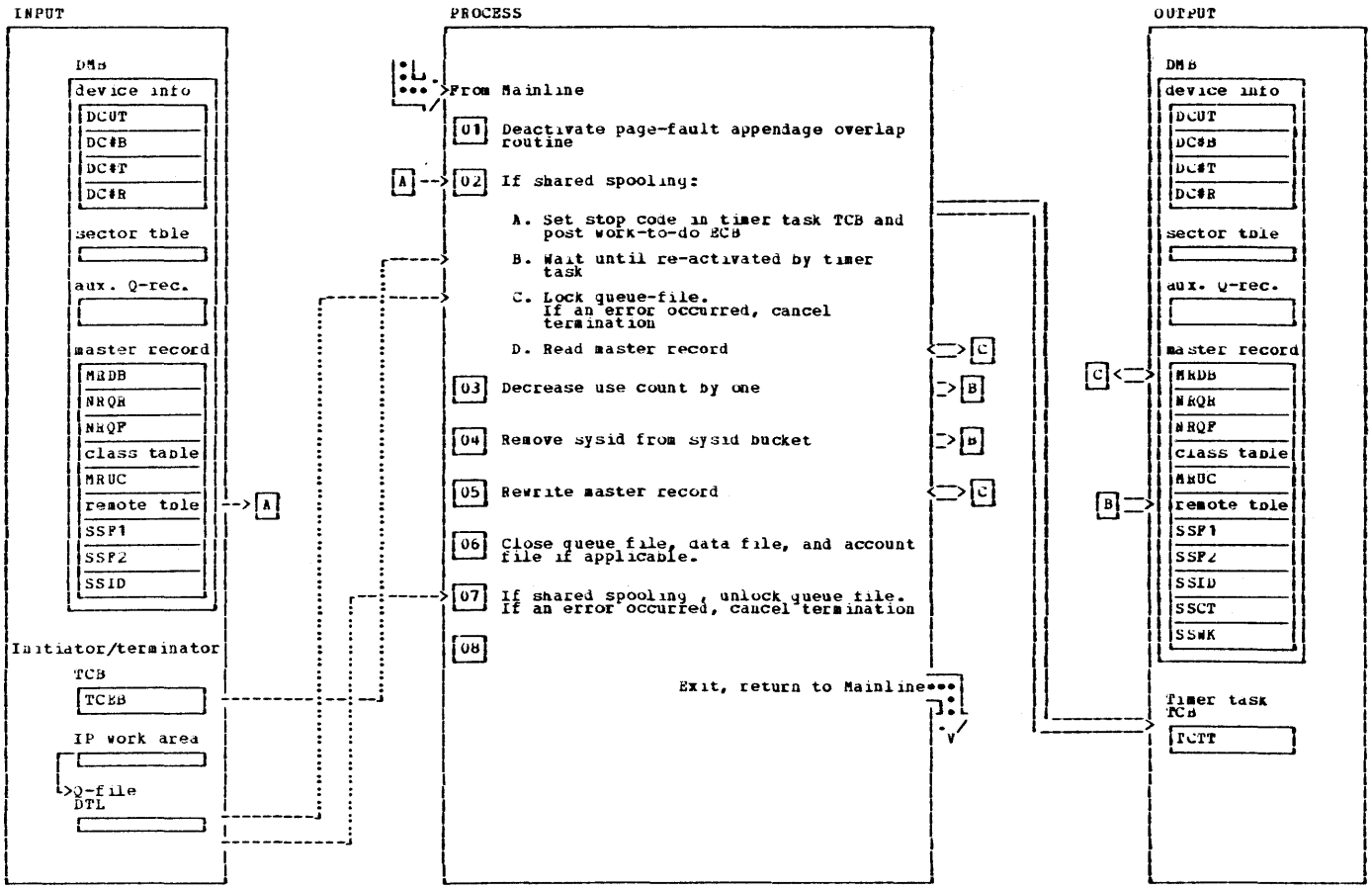
IPW$$T1 - Gather Accounting Information



NOTES	MODULE	LABEL	REF
Gather neccessary accounting information for final VSE/POWER accounting record from DOS/VSE partition-accounting table.		PX00	
- Storage is reserved for the execution account record being built			
- Information from the DOS/VSE accounting table is moved into the execution account record	COMREG		

NOTES	MODULE	LABEL	REF
- The DOS/VSE account-timer value is updated	GETJA		
- The current time is received and stored into the execution account record			$RDC
2 3 A null record (record length = 0) is written to indicate EOF to accounting			$PAM $PAR

IPW$ST1 - Final housekeeping



NOTES	MODULE	LABEL	REF
1 The page-fault-overlap routine is deactivated	SETPFA	T120	
The shared spooling function and the timer task are terminated			
2A The timer task is forced to complete, which detaches the DOS/VSE subtask and rewrites the DMB if necessary. The timer task then detaches itself.			
2B When the timer task has finished its processing (DOS/VSE subtask is detached), it reactivates the terminator task			$WFC
2C The queue-file must be acquired for exclusive use by this processor. This is done by issuing a LOCK request for the queue file. Return from the lock is only recieved when the lock is complete or when an error has occurred. In the later case message 1Q85I INTERNAL MACRO CALL FAILED, RC=rrmm is issued and VSE/POWER termination is canceled.	LOCK		

NOTES	MODULE	LABEL	REF
2D			$RDQ
3 The use count is decremented by one to reflect that this VSE/POWER system terminated properly.			
4 The sysid is removed from the sysid bucket, showing which systems are sharing the same queue file and data file			
5			$WIQ
6	CLOSER		
7 The queue file is unlocked to allow other processors to access the queue file	UNLOCK		
	CANCEL		$GAM

IPW$$XJ - VSE/POWER Scan Execution JECL Statement	
Label	Routine
XJ00	Function Entry
XJ18	Handle CTL, RDR and EOJ Statements
XJ30	Handle JOB Statement
XJ60	Handle LST and PUN Statements
XT00	Update Tasks
XA08	Process Positional Operands
XK00	Process Keyword Operands
XJ78	Handle SLI Statement
XJ86	Handle DATA Statement

Services Used	
Service	Macro
Task Management	IPW$ATT IPW$WFC IPW$WFI
Resource Management	IPW$RLR IPW$RSR
Storage Management	IPW$RLV IPW$RLW IPW$RSV IPW$RSW
Message Service	IPW$GAM IPW$WTO IPW$WTR

Functions used	
Module	Macro
IPW$$GD	IPW$GDR
IPW$$LU	IPW$ULP
IPW$$SL	IPW$GSL

Called by IPW$SXJ	
Module	Description
IPW$$XR	Execution Reader
IPW$$XW	Execution Writer

Labels	Chart XJ: 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.				
XJSD	The first 16 bytes constitute the section descriptor: 'XJCS release'				
	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$DQR - Queue record space			R5	
	6: IPW$DPD - Partition control block/Base register for subroutines			R6	
	7: **** - Addresses keyword being checked			R7	
	8: **** - Task control block (TNDS)			R8	
	9: **** - Second base register			R9	
	10: IPW$DPA - VSE/POWER nucleus			R10	
	11: IPW$DTC - Task control block			R11	
	12: **** - Reserved for nucleus use			R12	
	13: IPW$DSV - Task save area			R13	
	14: **** - Function linkage register			R14	
	15: XJCS - Function base register			R15	
	Note that the usage of registers 0-8 in the analysis routines starting at label XJ40 may differ from the above.				
	JECL Statment Analysis:				
XJ00	Save caller registers.				IPW$SAV
	Establish second base regiser using register 9			R9	
	Reserve space for a second function register save area.				IPW$RSw
	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 4 to locate the operation code. If no operation code is found, return to caller without processing the statement....>	XJ06		R4	
	Save the address of the operation code, which is contained in register 1, in register 7.			R1,R7	

Labels	Chart XJ: IPW$$XJ - Scan Execution JECL Statement		Modified Data Fields	Reg. Usage	Calls
XJ04	Match the operation code against the entries in the JECL statement routing table to determine which handling routine to branch to.				
	If a match is found, branch to the appropriate handling routine via table XJ12.....>	XJ12			
	If no match is found, return to caller without processing the statement.....>	XJ06			
	Return to Caller Without Having Processed the Statement:				
XJ06	If operator corrected JECL statement, branch to.....>	XJ10			
XJ07	Return the additional save area space to the storage pool.				IPW$RLW
	Return to IPW$$XR via link register 14, with return code 4 to process the statement.			R14	IPW$RET
	Return to Caller After Having Processed the Statement:				
XJ08	Branch and link to the 'bypass statement' routine.....>	XJ18		R14	
XJ10	If not operator corrected JECL, branch to.....>	XJ11			
	Release space acquired for card. Set up request word.		TCRW(IPW$DTC)		IPW$RLW
XJ11	If SLI in process, release parameter list space.				IPW$RLV
XJ11A	Return the additional save area space to the storage pool.				IPW$RLW
	Return to IPW$$XR via link register 14, with return code 0, to process the statement.			R14	IPW$RET
	JECL Statement Routing Table:				
XJ12	CTL.....>	XJ08			
XJJB	JOB.....>	XJ30			
	EOJ.....>	XJ24			
	RDR.....>	XJ08			
XJLT	LST.....>	XJ60			
XJPR	PRT.....>	XJ60			
XJPU	PUN.....>	XJ60			
XJSL	SLI.....>	XJ78			
	DATA.....>	XJ86			

Labels	Chart XJ: IPW$$XJ - Scan Execution JECL Statement		Modified Data Fields	Reg. Usage	Calls
	Handle CTL, RDR Statements:				
	Since CTL and RDR statements need not be processed at execution time, these statements are bypassed together with any of their continuation statements.				
XJ18	If there are no continuation cards, return to caller via link register 14			R14	
	Save return address and base register.			R14,R15 R0,R4	
	If the continuation card is for a writer-only partition, branch and link to the writer-only partition routine.....>	XS36			
XJ20	Get the continuation card.				IPW$GDR
XJ22	Restore the base register and the return address.			R14,R15 R0,R4	
	Branch to check for more continuation cards.....>	XJ18			
	Handle EOJ Statement:				
XJ24	If current SLI work space address is zero.....>	XJ26			
	Release SLI Workspace				IPW$GSL
XJ26	If the EOJ statement is not for a writer-only partition .....>	XJ08			
	Set job boundary switch to X'80'; clear job number field and indicator end of data on EOJ to ignore output till next $$JOB.		TCJB(IPW$DTC) PDJN(IPW$DPD) TCGP(IPW$LTC)		
	Set up register 4 for scanning the entries in the PDB to shut down any subordinate writer tasks.			R4	
XJ28	Address next entry. If end of table.....> If disposition not equal N.....>	XJ08 XJ2B			
	Set device class to L if printer or P if punch		TLCL(IPW$DDE)		
XJ29	Release ownership of device.....>	XJ28		R7	IPW$ULP
XJ2B	Check the entries in the PDB and if a task has been started:				
	• Reset the ownership in the device list entry in the PDB.		TLTC(IPW$DLE)		
	• Set the 'stop' termination type		TNTT(IPW$DTC)		
	• Post the event control block of the related task.		TNEB+2(IPW$DTC)		
	Wait for posting by the list task and..... >	XJ28			IPW$WFL

Labels	Chart XJ: IPW$$XJ - Scan Execution JECL Statement		Modified Data Fields	Reg. Usage	Calls
	Handle JOB Statement:				
	The content of the job statement is used to construct the job header record and the queue record for the ensuing job.				
XJ30	Set the job boundary switch to X'FF' to indicate job in progress.		TCJB(IPW$DTC)		
	Check for a record length of 72 or more. If the record length is longer and a writer only partition is serviced, it is set to 71. Branch and link to get the statement operand to.....>	XA00			
	* If there are no operands branch to.....>	XJ56			
	* If there are keyword operands branch to.....>	XK02			
	* If there are positional operands branch to.....>	XJ32			
XJ32	Load the address of variable 'JNM' for error message 1Q51I in register 7. Branch and link to scan the job name to.....>	XS24		R7 RE	
	Move the job name into the queue record and the job header record.		QRNM(IPW$LQR) NJHGNAM(IPW$DNR)	R4	
	If there are no more operands to be processed branch to.....>	XJ56			
	Branch and link to get next keyword.>	XS40			
	* If there is a next keyword..... >	XK02			
	* If end of statement..... >	XJ56			
	* If an error has been detected.... >	XT55		R1,R4	
	Job card parameter routing table:				
	JNM=.....>	XJ32			
	USER=.....>	XJ40			
	DISP=.....>	XJ52			
	PRI=.....>	XJ52			
	CLASS=.....>	XJ52			
	PWD=.....>	XJ50			
	SYSID=.....>	XJ4A			
	XDEST=.....>	PN00			
	LDEST=.....>	PN05			
	PDEST=.....>	PN10			
	NTFY=.....>	PN15			

Labels	Chart XJ: IPW$$XJ - Scan Execution JECL Statement		Modified Data Fields	Reg. Usage	Calls
	Process USER keyword:				
	The USER operand must be specified from one to sixteen alphameric characters, unless it is enclosed in single quotes.				
XJ40	Load the address of variable 'USER' for error message 1Q51I in reg. 7.			R7	
	Get the user information from the statement and move it into the queue record and into the job header record. If the user information has been specified incorrectly, issue message 1Q51I..... >	XT55	QRUI(IPW$DQR) NJHPUSER (IPW$DNR)	R1	
	Handle next operand..... >	XJ33			
	Process SYSID Keyword:				
	The SYSID parameter must be '1' to '9' or 'N'. If N is specified, the SYSID field in the Queue record is set to X'00'.				
XJ4A	Validate the operand.....>	XS60		R14	
	If nothing specified.....>	XJ4C			
	If the operand value is 'N', indicate no specific CPU wanted and branch ..>	XJ4C	QRSID(IPW$DQR)		
	If SYSID is not between '1' and '9', branch to report error.....>	XT85			
	Move SYSID value into Queue record.		QRSID(IPW$DQR)		
XJ4C	Move default/new SYSID value into job header record.		NJHPSYID (IPW$DNR)		
	Handle next operand..... >	XJ51			
	Process PWD Keyword:				
	The PwD must be specified as up to 8 alphameric characters.				
XJ50	Validate the operand.....>	XS60		R14	
	If nothing specified, branch to.....>	XJ501			
	Else move password into queue record.		QRPW(IPW$DQR)		
XJ501	Move default/new password into job header record.		NJHGPASS (IPW$DNR)	R4	
XJ51	Point to delimiter and branch to get next operand .....>	XJ33		R3	
XJ52	Point to delimiter and branch to get next operand ..... >	XJ33		R1,R3	
XJ56	If not writer only partition .....>	XJ08			
	If statement is to be logged, issue message 1Q47I.				
	Branch to.....>	XJ08			

Labels	Chart XJ: IPW$$XJ - Scan Execution JECL Statement		Modified Data Fields	Reg. Usage	Calls
	Process XDEST Keyword:				
	Valid formats are XDEST=* XDEST=nodeid XDEST=(*,userid) XDEST=(nodeid,userid)				
PN00	Load address of keyword verb XDEST in register 7 for error message 1Q51I and save registers 7 and 8.		SVR7(IPW$DSV)	R7	
	Load address of dummy field in register 7 and 8 and branch to.....>	PE00			
	Process LDEST Keyword:				
	Valid formats are LDEST=* LDEST=nodeid LDEST=(*,userid) LDEST=(nodeid,userid)				
PN05	Load address of keyword verb LDEST in register 7 for error message 1Q51I and save registers 7 and 8.		SVR7(IPW$DSV)	R7	
	Load address of target print node name in register 7 and address of target print remote name in register 8 and branch to.....>	PE00		R7,R8	
	Process PDEST Keyword:				
	Valid formats are PDEST=* PDEST=nodeid PDEST=(*,userid) PDEST=(nodeid,userid)				
PN10	Load address of keyword verb PDEST in register 7 for error message 1Q51I and save registers 7 and 8.		SVR7(IPW$DSV)	R7	
	Load address of target punch node name in register 7 and address of target punch remote name in register 8 and branch to.....>	PE00			
	Process NTFY Keyword:				
	Valid formats are NTFY=YES NTFY=(nodeid,userid)				
PN15	Load address of keyword verb NTFY in register 7 for error message 1Q51I and save registers 7 and 8.		SVR7(IPW$DSV)	R7	
	Load address of origin node name in register 7 and address of target user ID in register 8.			R7,R8	
	If not 'YES' specified, branch to.....>	PE00			

Labels	Chart XJ: IPW$$XJ - Scan Execution JECL Statement		Modified Data Fields	Reg. Usage	Calls
	Copy own node name from master record.		NJHGORGN (IPW$DNR)		
	Set userid to 'R000' to indicate that the notify message has to be routed to the local console.				
	Point to next delimiter end keyword and branch to.....>	PEEX		R1,R3	
PE00	If sublist notation, branch to .....>	PE20			
	If local node requested, copy local node name from master record, point to next delimiter and keyword and branch to.....>	PEEX		R1,R3	
PE10	Check if node ID specification starts with an alphabetic character and if all subsequent characters are in the range 'A-Z, 0-9, @, #, $' via branch and link to.....>	XS70			
	Move target node name addressed by register 7 and branch to.....>	PEEX			
PE20	If empty sublist, branch to.....>	XT85			
PE25	If local node requested, copy local node name from master record into field addressed by R7, update pointer and branch to .....>	PE40		R1,R3 R7	
PE35	Check if node ID specification starts with an alphabetic character and if all subsequent characters are in the range 'A-Z, 0-9, @, #, $' via branch and link to.....>	XS70			
	Move target node name and point behind delimiter			R3	
PE40	If second argument is omitted, branch to.....>	PEEX			
PE45	Check if second argument is alphameric by branch and link to.....>	XS60			
	If invalid delimiter, branch to.....>	XT85			
	If the user ID is specified as 1 to 3 digits, setup standard format 'RXXX' and branch to.....>	PE55			
PE50	Copy userid specification into field addressed by register 8.				
PE55	Point to next delimiter.			R1,R3	
PEEX	Reload saved registers 7 and 8 and process next keyword.....>	XJ33			

Labels	Chart XJ: IPW$$XJ - Scan Execution JECL Statement		Modified Data Fields	Reg. Usage	Calls
	Handle LST and PUN Statements: Task Initiation / Termination:				
XJ60	If partition canceling, then branch to.....>	XJ06			
	Reserve storage for the TCB of the new task.				IPW$RSW
	Set up register 8 as the base register for the new TCB.			R8	
	Set the storage ownership of the queue record to the execution reader TCB.				
	Initialize the storage descriptor with the information of the storage descriptor of the execution reader TCB.		TNSD(IPW$DTC)		
	Reserve storage for the queue records to be generated by the new task and set ownership to RDR TCB.				IPW$RSW
	Initialize the new queue record with system defaults and with information from the execution reader task:				
	• Copy body fields from the reader queue record		QNBF(IPW$DQR)		
	• Copy body extension fields		QNBZ(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$DQR)		
	Set up register 2 as base register for the disk management block.			R2	
	Check the statement to determine whether a LST or PUN queue record is to be formed. If a list record is to be formed branch to.....>	XJ64			
	Set punch defaults:				
	• Set record identifier to C'P' to indicate punch records.		QNQI(IPW$DQR)		
	• Insert punch default values.		QNSP(IPW$DQR)		
	Branch to.....>	XJ68			
XJ64	Set printer defaults:				
	• Set record identifier to C'L' to indicate list record.		QNQI(IPW$DQR)		
	• Insert printer default values.		QNSP(IPW$DQR)		
	• Records before split		QNBS(IPW$DQR)		
	• Records before message		QNBM(IPW$DQR)		
	• Additional count value		QNBN(IPW$DQR)		
	• Setup option byte		QNOP(IPW$DQR)		
	• Insert default line table		TNGW(IPW$DTC)		
	• Insert page size		QNER(IPW$DQR)		

Labels	Chart XJ: IPW$$XJ - Scan Execution JECL Statement		Modified Data Fields	Reg. Usage	Calls
XJ68	Set up register 4 for scanning the entries in the PDB.			R4	
	If segment macro has been issued, branch to.....>	XJ70			
	Set reader queue record identifier		QRQI(IPW$DGR)		
XJ70	Scan the entries in the PDB to locate the first printer or punch device defined.				
	In case of error, branch to issue message 1Q48I .....>	XT50			IPW$WTO IPW$WTC
XJ74	Move the physical device address in the queue record.		QNCU(IPW$DQR)		
	Reserve storage for the data set header record (DSHR). If no space available, go to exit.....>	XJ08			IPW$RSV
	Save DSHR address in the TCB		TN3E(IPW$DTC)		
	Initialize DSHR with defaults:		(All in IPW$DNR)		
	• Total DSHR length		NDHLEN		
	General Section of DSHR:				
	• General section length and ID		NDHGLEN/NDHGTYP		
	• Set modifier of general section		NDHGMOD		
	• Copy print/punch node and remote name from job header record		NDHGNODE NDHGRMT		
	• Copy node and remote name into queue record		QNTN,QNTU (IPW$DQR)		
	• Blank out some fields		NDHGPROC NDHGSTEP NDHGDD		
	• Copy output class from queue record		NDHGCLAS		
	• Assume variable record format		NDHGRCFM		
	• Set default record length to 512		NDHGLREC		
	• Copy data set copy count from queue record		NDHGSCT		
	• Set forms ID to blank		NDHGFCB		
	3800 Section of DSHR:				
	• 3800 section length and ID		NDHALEN/NDHATYP		
	• Insert modifier		NDHAMOD		
	• Set default flash count to 255		NDHAFLCT		

Labels	Chart XJ: IPW$$XJ - Scan Execution JECL Statement		Modified Data Fields	Reg. Usage	Calls
	VSE/POWER Section of DSHR:				
	• VSE/POWER section length and ID		NDHPLEN/NDHPTYPE		
	• Insert modifier		NDHPMOD		
	• Copy device type, priority, disp., user information, number of separators, option byte, partition id, target system id, compaction table name from queue record.		NDHPIDEV NDHPPRIO NDHPDISP NDHPNSEP NDHPOPTN NDHPPART NDHPSYID NDHPCOMP		
	SETPRT parameter list of DSHR:				
	• Indicate initialize printer		SPLFLAG1		
	• Set TRC=NO and DEBUG=NORM		SPLFLAG2		
	• Set flash count to 255		SPLFLSHC		
	• Set first copy group to one		SPLCOPYG		
	• Set copy group index to 1		SPLCINDX		
	• Set length of parameter list		SPLLNGTH		
	• Set default required flag		TNF3(IPW$DTC)		
	Branch and link to the statement analysis routine.....>	XA00			
	If there are no operands branch .....>	XT00			
	If there are keyword operands branch>	XK02			
	If there are positional operands branch to.....>	XA08			

Labels	Chart XJ: IPW$$XJ - Scan Execution JECL Statement		Modified Data Fields	Reg. Usage	Calls
	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 equal to first value (QNBM).		QNBN(IPW$DQR)		
XT05	Copy the device address from the queue record into the storage descriptor of the new TCB to determine which execution writer task, if any, the new task will replace.		TNCU(IPW$DTC)		
	Set up register 4 for scanning the entries in the PDB.			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.				
XT15	If DISP=N was specified for the related entry, release ownership of the PUB specified by that entry.				IPW$ULP
XT20	Save the address of the entry in the new TCB.		TNR4(IPW$DTC)		
	Set the device type.		QNDT(IPW$DQR)		
	If no subordinate task is running or the reader is task owner branch to.....>	XT25			
	Terminate the subordinate task: • Set 'stop' termination type code. • Post event control block. • Exit to task selection.		TNTT(IPW$DTC) TNEB+2(IPW$DTC)		IPW$WFC
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 and records before segmentation to zero and the transmission count to one.		QNNC(IPW$DQR) QNCG(IPW$DQR) QNBS(IPW$DQR) QNTC(IPW$DQR)		
XT27	Check if the disposition is I and, if so, test if this is punch output. If not, change disposition to D and branch to.....>	XT30	QNDP(IPW$DQR)		

Labels	Chart XJ: IPW$$XJ - Scan Execution JECL Statement		Modified Data Fields	Reg. Usage	Calls
XT28	Set job suffic number to zero.		QNSN(IPW$DQR)		
	Load adress of the master record in register 3 and reserve the disk management block			R3	IPW$RSR
	Increment current job number by one and test for overflow. If overflow, set job number to one. Save new value in the disk management block		MRNO(IPW$DQC)	R2	
	Release disk management block				IPW$RLR
XT30	The data set header record is now updated with values from queue record in the following fields:		(All in IPW$DNR)		
	• Target node name		NDHGNODE		
	• Target user ID		NDHGRMT		
	• Output class		NDHGCLAS		
	• Copy count		NDHGDSCT		
	• Forms identifier		NDHGFORM		
	• Forms overlay ID		NDHAFLSH		
	• Copy groups		NDHACPYG		
	• Device type		NDHPIDEV		
	• Output priority		NDHPPRIO		
	• Disposition		NDHPDISP		
	• User information		NDHPUSER		
	• Number of separators		NDHPNSEP		
	• Option byte		NDHPOPTN		
	• Target SYSID		NDHPSYID		
	• Compaction table name		NDHPCOMP		
	• Password		NDHPPASS		
	Attach the new writer task into the system.				IPW$ATT
	Return to IPW$$XR.....>	XJ10			
XT35	Get ownership.				IPW$ULP
	If no ownership is received:				
	• Set disposition to C'D'.		QNDP(IPW$DQR)		
	• Issue message 1Q46I.				IPW$GAM
	• Branch to.....>	XT25			
XT40	Set the device type code in the device list entry in the PDB to C'N' to indicate no interception of requests for this device.		TLCL(IPW$DTL)		
XT45	If no TCB present, branch to.....>	XJ10			
	Release the queue space.			R1	IPW$RLW
	Release the data set header record space.			R1	IPW$RLV
	Release TCB storage.			R1	IPW$RLW
	Exit.....>	XJ10			

Labels	Chart XJ: IPW$$XJ - Scan Execution JECL Statement		Modified Data Fields	Reg. Usage	Calls
	Error message handling				
XT50	Load address of message 1Q48I in register 14 and branch to.....>	XT86		R14	
XT55	Load address of message 1Q49I in register 14 and branch to.....>	XT86		R14	
XT60	Load address of message 1Q50I in register 14 and branch to.....>	XT86		R14	
XT65	Load address of 'DISP' keyword verb in register 7 and branch to.....>	XT85		R7	
XT70	Load address of 'CLASS' keyword verb in register 7 and branch to.....>	XT85		R7	
XT75	Load address of 'LST' keyword verb in register 7. If the current queue record is a list queue record, branch Otherwise load address of 'PUN' keyword verb in register 7 and branch	XT85 XT85		R7	
XT80	Load address of 'LTAB' keyword verb in register 7 and branch.....>	XT85		R7	
XT85	Load address of message 1Q51I in register 14.			R14	
XT86	If SLI in process, release parameter list space and reset SLI indication.		TCF2(IPW$DTC)		IPW$RLV
XT87	Load address of partition control block in register 6 and address of erroneous card in register 4, set up message request word and write JECL statement that is in error on SYSLOG.		TCMW(IPW$DTC)	R6 R4	IPW$WTO
	Write appropriate error message.				
	If the incorrect statement is in a continuation card, branch to.....>	XT88			
	If the incorrect statement is a 'LST', 'PRT', 'PUN' or 'JOB' or 'SLI' statement, branch to.....>	XT90			
XT88	Write message '1R33D' on SYSLOG.				IPW$GAM
	Bypass any continuation card: branch and link to.....>	XJ18		R14	
	Branch to.....>	XJ92			
XT90	Write message '1R33D' on SYSLOG				IPW$GAM
XT92	If a TCB has not been acquired, branch to.....>	XT94			
	If no DSHR space exists, branch.....>	XT93		R1	
	Release DSHR space			R1	IPW$RLV
XT93	Release queue and TCB space				IPW$RLW

Labels	Chart XJ: IPW$$XJ - Scan Execution JECL Statement		Modified Data Fields	Reg. Usage	Calls
XT94	If error statement from operator, branch to.....>	XT96			
	Reserve space for correction.				IPW$RSW
XT96	Set up space for operator correction.				
	Write a blank and ask for operator correction.				IPW$WTR
	If operator replied EOB, branch to..>	XJ10			
	If FLUSH entered, branch to.....>	XT98			
	If the corrected card is "* $$ LST" "* $$ PRT" or "* $$ PUN" or "* $$ JOB" or "* $$ SLI", branch ...>	XJ02			
XT97	Write message 1R33I and branch .....>	XT90			IPW$GAM
XT98	Set flush cancel code.		TCTT(IPW$DTC)		
	Set off dump option.		CMRG		
	Set off partition dump option				
	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	
XA00	The statement is checked to determine whether there are:				
	• No operands, return.....>	0(R14)			
	• Keyword operands, return.....>	4(R14)			
	• Positional operands, return.....>	8(R14)			
	If the statement is in error .....>	XT55			

Labels	Chart XJ: IPW$$XJ - Scan Execution JECL Statement		Modified Data Fields	Reg. Usage	Calls
	Process Positional Operands:				
XA08	First Positional Operand:				
	This operand represents the class and attributes to be assigned to the queue entry.			R0,R1 R3,R7	
	If the disposition attribute is specified, branch to disposition validation subroutine ....>	XS30			
	If tape support, set no segmentation and indicate tape.		QNBS(IPW$DQR) TNSI(IPW$DTC)		
	If punch output, branch to.....>	XA09A			
	If disposition "I" has been specified for list queue entry branch to issue error message.....>	XT65			
XA09A	The disposition indicator is reset.		QNDP(IPW$DQR)		
	If the class attribute is specified, reset 'disposition/class' indicator.		QNCL(IPW$DQR)		
	In case of error, branch to issue message 1Q51I. .....>	XT70			
XA10	Get 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. If the operand is specified, the 'forms' indicator is reset.		QNFI(IPW$DQR)	R0,R1, R2,R3, R7	
	In case of error, branch to issue message 1Q51I. .....>	XT85			
XA18	The data set header record is addressed using register 4 as base. and the form number saved.		SPLFORMS (IPW$DTE)	R4	
	Get next operand. .....>	XS40			
	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. If a tape unit is specified, the address of the variable 'TADDR' is loaded into register 7 and a branch and link is made to the tape subroutine.....>	XS28		R2,R4, R7 R3,R7	
	Get next operand.....>	XA22			

Labels	Chart XJ: IPW$$XJ - Scan Execution JECL Statement		Modified Data Fields	Reg. Usage	Calls
XA20	If the number of copies is specified, the address of the variable 'COPY' is loaded into register 7 and a branch and link is made to the numeric operand processing subroutine to convert the operand.....>	XS00		R2,R3, R4 R14	
	The 'number of copies' indicator is reset.		QNNC(IPW$DQR)	R2	
XA22	Branch and link to get the next operand.....>	XS40		R14	
	Fourth Positional Operand:				
	This operand represents the number of output records to be handled before the warning message 1Q52I is issued to the operator.			R2,R4, R7	
	If the operand is specified, the address of the variable 'RBM' is loaded into register 7 and a branch and link is made to the numeric operand processing subroutine to convert the operand.....>	XS00		R7 R14	
	The RBM indicator is reset.		QNBM(IPW$DQR)		
	Branch and link to get the next operand.....>	XS40		R14	
	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			
	Process Keyword Operands:				
XK00	Branch and link to get the next operand.....>	XS40		R14	
XK02	Load the address of the appropriate keyword routing table into register 1. (JOB statement keyword table, if a JOB statement is processed, LST/PUN keyword routing table else)			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 found, branch to issue error message 1Q50I.....>	XT60			IPW$WTC
	Branch to the corresponding operand handling routine.			R6	

Labels	Chart XJ: IPW$$XJ - Scan Execution JECL Statement		Modified Data Fields	Reg. Usage	Calls
	Subroutines:				
	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$WTC
	Convert the value of the field to binary in register 2.			R2	
	Return to caller via register 14.				
	Line Table Entry Subroutine:				
XS10	On entry to this routine, register 3 contains the address of the first byte of the field to be scanned.			R3	
	Scan the operand field to insure that it consists of numeric characters only and that it has a length of 26. If the field is correct, each duplet is converted to binary and stored in the line table field of the TCB.		TCGW(IPW$DTC)		
	In case of error, branch to issue message 1Q51I INVALID LTAB PARAMETER..... >	XT80			IPW$WTO
	Return to caller via register 14.			R14	
	Alphameric Phase Name Subroutine:				
XS24	On entry to this routine, register 3 contains the address of the first byte of the phase name.			R3	
	Scan the phase name to ensure that it consists of alphameric characters.				
	In case of error, branch to issue the error message, the address of which is contained in register 7..... >	XT85		R7	IPW$WTC
	Get the machine length of the phase name in register 4.			R4	
	Return to caller via register 14.			R14	
	Tape Subroutine:				
XS28	Check the tape unit address in register 3 for hexadecimal characters. If it is invalid, branch to issue the error message, the address of wich is contained in register 7..... >	XT85		R7 R7	IPW$WTO
	Move the tape address to the new TCB.		TNPU+1(IPW$DTC)		
	Indicate tape support.		TNSI(IPW$DTC)		
	Return to caller via register 14.			R14	

Labels	Chart XJ: IPW$$XJ - Scan Execution JECL Statement		Modified Data Fields	Reg. Usage	Calls
	Disposition Validation Subroutine:				
XS30	Get the address of the disposition table in register 1.			R1	
	Match the disposition against the entries in the table. If an invalid disposition was specified, branch to issue error message 1Q51I INVALID DISP PARAMETER..... >	XT65			IPw$wTO
	Return to caller via register 14.			R14	
	Get Continuation Card for Writer-only Partition:				
XS36	If a continuation card is to be read for a writer-only partition, a return is made to IPW$$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$DTC)		
	Restore registers 14 through 5.			R14-R5	
	Return to caller via register 14.			R14	
	Get Next Operand Subroutine:				
XS40	On entry to this routine, register 3 contains the address of the field delimiter of the current (processed) operand.			R3	
	Return in made via register 14: • If a next operand is present.... > • If end of statement..... > • If an error has been detected... >	 0(R14) 4(R14) 8(R14)			
	If the remaining operands to be processed are on a continuation card, read the continuation card.				IPw$GDR
	Load the address of the new operand into register 3.			R3	
	Return to caller via register 14.			R14	

Labels	Chart XJ: IPW$$XJ - Scan Execution JECL Statement		Modified Data Fields	Reg. Usage	Calls
	Validate Operand 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	
XS70	Alphameric Operand Subroutine:				
	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 permissable length of the field.			R3,R4	
	Scan the operand field; the first character must be alphabetic. Valid characters in subsequent positions are A-Z, 0-9, @, # and $. If the operand contains other characters, branch to.....>	XT85		R1,R2	
	Examine the delimiter. If a comma (,) or a blank ( ) or a closing paren-this, branch to.....>	XS72		R1	
	If SLI in process and the delimits in a point (.), branch to..... >	XS72			
	If not stopped by end of statement, branch to.....>	XT85		R0	
XS72	Get machine length of operand in register 4.			R4	
	Return to caller via register 14.			R14	

Labels	Chart XJ: IPW$$XJ - Scan Execution JECL Statement		Modified Data Fields	Reg. Usage	Calls
	Handle SLI Statement:				
XJ78	If SLI is not supported branch .....>	XJ84			
	Reserve parameter list space. If not successful, branch .....>	XJ08		R1	IPW$RSV
	Indicate SLI in process		TCF2(IPW$DTC)		
	Blank character fields of parameter list		XJPMER,XJPTYP XJPUID,XJPPWD		
	Branch and link to get operands ....>	XA00		RE	
	• If no operands present .....>	XJ08			
	• If keyword operands .....>	XJS00			
	• If positional operands .....>	XJS20			
XJS00	Scan the keyword table to determine the appropriate handling routine			R1,R4	
	MEM= .....>	XJS30			
	ICCF=.....>	XJS40			
	LIB= .....>	XJS60			
XJS10	Copy originating user from queue rec.		XJPUID		
	Load parameter list address into register 1 and branch and link to IPW$$SL ..... >	XJ10		R1 RE,RF	IPW$GSL
	Positional Member Operand:				
XJS20	If sublib is specified, move it into the parameter list, else use the default from the generation.				
	If member name is invalid or omitted, branch to issue message 1Q51I, ...>	XT85		R4,RE	
	Move member name into parm. list ...>	XJS10			
	Librarian Member Operand (MEM=):				
XJS30	If a member has already been processed or ICCF= has been specified or the specification of MEM= is invalid, branch .....>	XT85			
	Move member name and type into the parameter list, branch .....>	XJS10	XJPMBR,XJPTYP		

Labels	Chart XJ: IPW$$XJ - Scan Execution JECL Statement		Modified Data Fields	Reg. Usage	Calls
	ICCF Member Operand (ICCF=):				
XJS40	If the keyword is a sublist, branch >	XJS50			
	If member name is invalid or omitted, branch to issue message 1Q51I, branch ..... >	XT85			
	Move member name into parameter list		XJPMBR		
	If end of statement, branch ..... >	XJS45			
	If next delimiter is invalid, branch >	XT85			
	Point to next keyword, branch ..... >	XJS00			
XJS45	If no LIB=keyword has been processed, branch to issue message 1Q51I, ... >	XT85			
	Otherwise, branch ..... >	XJS10			
XJS50	If member name is invalid or omitted, branch to issue message 1Q51I, ... >	XT85			
	Move member name in to parameter list		XJPMBR		
	If next delimiter is a comma, branch >	XJS55			
	If next delimiter is not a ')', branch to issue message 1Q51I, ... >	XT85			
	If the end of statement is reached .>	XJS45			
	Point to next keyword, branch ..... >	XJS00			
XJS55	If the password is invalid or omitted or the next delimiter is not ')', branch to issue message 1Q51I ..... >	XT85			
	Move password to parameter list		XJPPWD		
	If the end of statement is reached .>	XJS45			
	Point to next keyword, branch ..... >	XJS00			
	Process Sublibrary Specification (LIB=):				
XJS60	If the keyword is in sublist format >	XJS70			
	Store binary sublib value into parameter list				
	If the end of statement is reached .>	XJS65			
	If delimiter is not ',' branch to issue message 1Q51I, ... >	XT85			
	Point to next keyword, branch ..... >	XJS00			
XJS65	If ICCF= keyword was not processed, branch to issue message 1Q51I..... >	XT85			
	Otherwise, branch ..... >	XJS10			

Labels	Chart XJ: IPW$$XJ - Scan Execution JECL Statement		Modified Data Fields	Reg. Usage	Calls
XJS70	Check the sublib values. If invalid or an invalid delimiter is found, branch to issue message 1Q51I.....	XT85			
	If the end of statement is reached ..	XJS45			
	Point to next keyword, branch	XJS00			
XJ84	Issue message 1Q45I.				IPW$GAM
	Set termination type in the TCB to C'F'.		TCTT(IPW$DTC)		
	Return to IPW$$XR.....	XJ08			
	Handle DATA Statement:				
XJ86	If no SLI workspace is available, return to IPW$$RR.....	XJ06			
	Reset reader switch to C'R' in the SLI work space.		SLRR(IPW$LSL)		
	Return to IPW$$XR.....	XJ08			

Labels	Chart XJ: IPW$$XJ - Scan Execution JECL Statement		Modified Data Fields	Reg. Usage	Calls
	Process Keyword Operands:				
	Disposition Operand (DISP=):				
XK10	Branch and link to the disposition validation subroutine.....>	XS30		R14	
	If it in punch output, branch to.....>	XK11			
	If disposition "I" has been specified for list queue entry, branch to issue error message.....>	XT65			
	Reset the disposition indicator.		QNDP(IPW$DQR)		
	If disposition is tape, reset segmentation and indicate tape support.		QNBS(IPW$DQR) TNSI(IPW$DTE)		
	Branch to get the next operand and process it.....>	XK00			
	Class Operand (CLASS=):				
XK12	In case of error, branch to issue message 1Q51I.....>	XT70			
	Reset the class indicator.		QNCL(IPW$DQR)		
	Branch to get the next operand and process it.....>	XK00			
	Remote Operand (REMOTE=):				
XK14	Load the address of variable 'REMCTE' into register 7 and branch and link to the numeric operand processing subroutine to convert the operand...>	XS00		R7 R14	
	If in error, branch to.....>	XT85		R14	
	Reset remote indicator		QNTJ(IPW$DQR)		
	Convert remote ID to character representation "RXXX" and save it in target user field.		QNTU(IPW$DQR)		
	Copy local node name from master record		QNTN(IPW$DQR)		
	Branch to get next operand and process it.....>	XK00			
	Forms Number Operand (FNO=):				
XK16	Load the address of variable 'FNC' into register 7 and branch and link to the validate operand subroutine.....>	XS60		R7	
	Reset the forms number identifier if specified.		QNFI(IPW$DQR)		
	Branch to get next operand and process it.....>	XK00			

Labels	Chart XJ: IPW$$XJ - Scan Execution JECL Statement		Modified Data Fields	Reg. Usage	Calls
	Job Separator Operand (JSEP=):				
XK24	Set the maximum permissible length in register 4.			R4	
	Load the address of keyword verb 'JSEP' 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 seperator 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$DQR)		
	If 'Y' is specified, branch to.....> If 'N' is specified, set suppress pages between copies flag in QR. Otherwise, branch to.....>	XK2C XT85	QNOP(IPW$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 XJ: IPW$$XJ - Scan Execution JECL Statement		Modified Data Fields	Reg. Usage	Calls
	Copies Operand (COPY=):				
XK26	Load the address of keyword verb 'COPY' for message 1Q51I into register 7 and branch and link to the numeric operand processing subroutine to convert the operand.....>	XS00		R4,R7	
	Reset the number of copies indicator.		QNNC(IPW$DQR)		
	Branch to get the next operand and process it.....>	XK00			
	Tape Address Operand (TADDR=):				
	The TADDR may be defined as 3 hexadecimal numbers specifying the tape address, or as the same numbers enclosed by X'', indicating hex valued. Mode may be optionally specified.				
XK28	Load address of variable 'TADDR' into R7. If the operand starts with a parenthesis, update pointer by 1.			R7	
XK30	If the operand is not in hex format, branch to.....>	XK32			
	Point to tape address and check validity.....>	XS28		R14	
	If no ending quote, branch to error.>	XT85			
	Else branch to.....>	XK34			
XK32	Check tape address for validity.....>	XS28		R14	
	Update operand pointer.			R3	
XK34	If mode is not specified, branch to.>	XK38			
	If there is no comma delimiter, branch to.....>	XT85			
	If not in hex format, branch to.....>	XK35			
	Check the mode specification.....>	XK3A		R14	
	Point past ending quote, branch to..>	XK36			
XK35	Check mode specification.....>	XK3A		R14	
XK36	If no ending parenthesis, branch to.>	XT85			
XK38	Set disposition to tape and reset segmentation.		QNDP(IPW$DQR) QNBS(IPW$DQR)		
	Branch to.....>	XK00			
XK3A	Translate the mode and pack to internal format. Move into TCB.		TNTM(IPW$DTC)		
	Return to caller.....>	R14			

Labels	Chart XJ: IPW$$XJ - Scan Execution JECL Statement		Modified Data Fields	Reg. Usage	Calls
	Records before Message Operand (RBM=):				
XK40	Reset the records before message indicators.		QNBM(IPW$DQR) QNBN(IPW$DQR)		
	In case of error, branch to issue message 1Q51I invalid RBM parameter.....>	XT85			
	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 keyword verb 'RBS' for message 1Q51I into register 7 and branch and link to the numeric operand processing subroutine to convert the operand.....>	XS00		R4,R7	
	Reset the records before split indicator if not tape spooling. Branch to get the next operand and process it.....>	XK00	QNBS(IPW$DQR)		
	UCS Buffer Operand (UCS=):				
XK52	Load the address of keyword verb 'UCS' for message 1Q51I into register 7 and branch and link to the alphameric phas name subroutine.....>	XS24		R7	
	Move the phase name to the data set header record		NDHGUCS(IPW$DNR)		
	If the FOLD or CHECK option was specified, the option is set in the data set header record		NDHGUCSO (IPW$DNR)		
XK65	If the CHARS operand is not specified, the UCS specification is treated as if the UCS image specification has been specified for CHARS.			R4	
	The SETPRT parameter list is addressed using register 4 as base. If CHARS is specified, branch to.....>	XK00		R4	
	Otherwise, move the UCS phase name into the SETPRT parameter list.		SPLCHAR1(SPLIST)		
	Branch to get the next operand and process it.....>	XK00			

Labels	Chart XJ: IPW$$XJ - Scan Execution JECL Statement		Modified Data Fields	Reg. Usage	Calls
	FCB Buffer Operand (FCB=):				
XK72	Load the address of keyword verb 'FCB' for message 1Q51I into register 7 and branch and link to the alpha-meric phase name subroutine.....>	XS24		R7	
	Move the phase name into the data set header record.		NDHGFCB(IPW$DNR)		
	The SETPRT parameter list is addressed using register 4 as a base.			R4	
	The fifth through eighth character of FCB name is moved into the SETPRT parameter list.		SPLFCB(SPLIST)		
	Branch to get the next operand and process it.....>	XK00			
	List Device Operand (LST= or PUN=):				
XK80	In case of error, branch to issue message 1Q51I. .....>	XT75		R2,R3, R4,R7	
	Check for valid device.				IPW$ULP
	Store the channel and unit number.		QNCU(IPW$DQR)		
	Branch to get the next operand and process it.....>	XK00			
	Priority Operand Handler (PRI=):				
XK94	In case of error, branch to issue message 1Q51I.....>	XT85			
	Check parameter for numeric.....>	XS00		R4,R7	
	Store priority parameter.		QNPY(IPW$DQR)		
	Branch to get next operand and process it.....>	XK00			
	Compact Operand Handler (CMPACT=):				
XK96	In case of error, branch to issue message "1Q51I INVALID COMPACT PARAMETER."....>	XT85		R7	
	Check parameter for alphameric and first character only alphabetic.....>	XS70		R3	
	Move the compact name.		QNCP(IPW$DQR)	R3	
	Branch to get next operand and process it.....>	XK00			

Labels	Chart XJ: IPW$$XJ - Scan Execution JECL Statement		Modified Data Fields	Reg. Usage	Calls
	Flash Operand (FLASH=):				
XL00	Load the address of keyword verb 'FLASH' for message 1Q51I into register 7.			R7	
	If opening parenthesis, branch to...>	XL05			
	Branch to validate the operand.....>	XS60			
	Reset the flash identifier and set maximum flash count of 255.		QNFL(IPW$DQR) SPLFLASH(SPLIST) SPLFLSHC(SPLIST)		
	Branch to get next operand.....>	XK00			
XL05	Branch to validate the operand.....>	XS60		RE	
	Reset the flash identifier and set maximum flash count of 255.		QNFL(IPW$DQR) SPLFLASH(SPLIST) SPLFLSHC(SPLIST)		
XL07	If the delimiter is not a comma branch to.....>	XL10		R1	
	Address second value (flash count).			R3	
	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(SPLIST) NDHAFLCT (IPW$DNR)		
XL10	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 keyword verb 'BURST' for error message '1Q51I' into register 7.			R7	
	Assume BURST=N is specified.		SPFLAG1(SPLIST)		
	Examine operand:				
	If 'Y' is specified, branch to.....> If 'N' is specified, branch to.....>	XL16 XL18			
	Otherwise, it is an error, branch to.....>	XT85			

Labels	Chart XJ: IPW$$XJ - Scan Execution JECL Statement		Modified Data Fields	Reg. Usage	Calls
XL16	Set burst flag in queue record.		QNPS(IPW$DQR)		
	Set burst flag in the 3800 section of the data set header record.		NDHAFLG1 (IPW$DNR)		
	Set burst flag in the SETPRT parameter list of the data set header record.		SPLFLAG1(SPLIST)		
XL18	Address next possible operand.			R3	
	Branch to get next operand.....>	XK00			
	DFLT Operand Handler (DFLT=):				
XL20	Load address of keyword verb 'DFLT' for error message '1Q51I' in register 7.				
	Examine operand:				
	If 'Y' is specified, branch to.....>	XL22			
	If 'N' is not specified, branch to..>	XT85			
	Indicate no default requested.		TNF3(IPW$DTC)		
XL22	Address next possible operand.			R3	
	Branch to get next operand.....>	XK00			
	COPYG operand (COPYG=):				
	Load address of keyword verb 'COPYG' for error message '1Q51I' into register 7.			R3	
	If operand is enclosed in parentheses, branch to.....>	XL32			
	Branch to validate and convert.....>	XS00		RE	
	If nothing is specified or value greater than 255, branch to.....>	XT85		R2	
	Store copy group index in queue record.		QNCG(IPW$DQR)		
	Set transmission count to one.		QNTC(IPW$DQR)		
	Branch to.....>	XL39			
XL32	Branch to validate and convert.....>	XS00		RE	
	If value is zero, or value greater than 255, or more than 8 copy groups are specified, branch to.....>	XT85		R2,R4	
	Store copy group in queue record and address next sub operand.		QNCG(IPW$DQR)	R2	
XL38	If no text suboperand, calculate total value of all copy groups specified.				
	If total value greater than 155, branch to.....>	XT85			

Labels	Chart XJ: IPW$$XJ - Scan Execution JECL Statement		Modified Data Fields	Reg. Usage	Calls
XL39	Update the SETPRT parameter list of the data set header record		SPLCOPYG(SPLIST)		
	Branch to get next operand.....>	XK00			
	MODIFY Operand (MODIFY=):				
XL40	Load the address of the keyword verb 'MODIFY' for error message '1Q51I ..' into register 7.			R7	
	If opening parenthesis, branch to...>	XL42		R3	
	Branch to validate operand.....>	XS60		RE	
	If nothing is specified, branch to..>	XT85		R4	
	Update the copy modification field in the 3800 section of the data set header record and in the SETPRT parameter list of the VSE/POWER section of the data set header record.		NDHAMODF (IPW$DNR) SPLCPMOD(SPLIST)	R4 R4	
	Branch to get next operand.....>	XK00			
XL42	Bypass the opening parenthesis and set the maximum permissible length in register 4.				
	Branch to validate the first specification.....>	XS60		RE	
	If nothing is specified, branch to..>	XT85		R4	
	Update the copy modification field in the 3800 section of the data set header record and in the SETPRT parameter list of the VSE/POWER section of the data set header record.		NDHAMODF (IPW$DNR) SPLCPMOD(SPLIST)	R4 R4	
	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	
	Update character arrangement table in the SETPRT parameter list of the data set header record.		SPLCMCHR(SPLIST)	R4	
XL43	If the delimiter is not a closing parenthesis, branch to.....>	XT85		R1	
	Branch to get next operand.....>	XK00			

Labels	Chart XJ: IPW$$XJ - Scan Execution JECL Statement		Modified Data Fields	Reg. Usage	Calls
	CHARS Operand (CHARS=):				
XL50	Load the address of the keyword verb 'CHARS' for error message into register 7.			T7	
	If opening parenthesis, branch to...>	XL54			
	Branch to validate operand.....>	XS60		RE	
	If nothing is specified, branch to..>	XL52		R4	
	Move the character arrangement table into the 3800 section of the data set header record and into the SETPRT parameter list of the VSE/POWER section of the data set header record.		NDHATAB1 (IPW$DNR) SPLCHAR1(SPLIST)		
XL52	Address delimiter and branch to.....>	XL56			
XL54	Save first translate table field address and first character arrangement table field address in save area.		SVR0(IPW$DSV) SVR1(IPW$DSV)		
XL55	Branch to validate operand.....>	XS60			
	If nothing specified, branch to.....>	XT85			
	If more than four CATs are specified, branch to.....>	XT85			
	Load address of current translation table field in register 14 and move CAT value.		NDHATAB1 (IPW$DNR)	R14	
	Bump to next translation table field and save address.		SVR0(IPW$DSV)		
	Load address of current CAT field in register 14 and move character arrangement table.		SPLCHAR1 (IPW$DNR)	R14	
	Bump to next CAT fiels and save address.		SVR1(IPW$DSV)		
	Increment number of processed CATs.		SVR0	R14	
	Point register 3 to delimiter.			R3	
	If continuation, branch to.....>	XL55			
	If not closing parenthesis, branch to.....>	XT85			

Labels	Chart XJ: IPW$$XJ - Scan Execution JECL Statement		Modified Data Fields	Reg. Usage	Calls
XL56	Examine if the UCS operand is specified. If so, branch to.....>	XK00			
	Otherwise, take the first CHARS value Branch to get next operand.....>	XK00	NDHGUCS(IPW$DNR)		
	User Operand (USER=):				
XL60	Check the USER operand for valid characters and maximum length of 16.				
	If in error issue message 1Q51I and branch to.....>	XT85			
	Else move user information into queue record.		QNUI(IPW$DQR)		
	Get next operand.....>	XK00			
	Password Operand (PWD=):				
XL70	Validate the password. Must be alphameric and not greater than 8 characters.....>	XS60		R14	
	If in error, branch to issue message 1Q51I.....>	XT85			
	Move password into queue record.		QNPW(IPW$DQR)		
	Get next operand.....>	XK00			
	SYSID Operand (SYSID=):				
	The SYSID operand must be specified as 1 to 9 or 'N'. If 'N' is specified, then the target SYSID in the queue record will be set to hex zero.				
XL90	Validate the operand.....>	XS60		R14	
	If in error issue message 1Q51I and branch to.....>	XT85			
	If the operand is 'N', move hex zero into the queue record.		QNSID(IPW$DQR)		
XL93	Else move the SYSID into the queue record.		QNSID(IPW$DQR)		
XL96	Branch to get next operand.....>	XK00			
	DEST Operand (DEST=): Valid formats are DEST=* DEST=nodeid DEST=(*,userid) DEST=(nodeid,userid)				

Labels	Chart XJ: IPW$$XJ - Scan Execution JECL Statement		Modified Data Fields	Reg. Usage	Calls
PP00	Load the address of the keyword verb 'DEST' for error message '1Q51I ..' in register 7.			R7	
	If sublist notation, branch to.....>	PP20			
	If local node requested, copy local node name from master record, point to next delimiter and process next keyword.....>	XK00	QNTN(IPW$DQR)	R3	
PP10	Check if nodeid specification starts with an alphabetic character and if all subsequent characters are in the range 'A-Z, 0-9, @, #, $' via branch and link to.....>	XS70		R14	
	Move target node name and process next keyword.....>	XK00	QNTN(IPW$DQR)		
PP20	If empty sublist, branch to.....>	XT85			
PP25	If local node requested, copy local node name from master record, update pointers and branch to.....>	PP40	QNTN(IPW$DQR)	R1,R3	
PP35	Check if nodeid specification starts with an alphabetic character and if all subsequent characters are in the range 'A-Z, 0-9, @, #, $' via branch and link to.....>	XS70		R14	
	Move target node name and point behind delimiter.		QNTN(IPW$DQR)	R3	
PP40	If second argument is omitted, branch to process next keyword.....>	XK00			
PP45	Check if second argument is alpha-meric by branch and link to.....>	XS60		R14	
	If invalid delimiter, branch to.....>	XT85			
	If the userid is specified as 1 to 3 digits, setup standard format 'RXXX' and branch to process next keyword..>	XK00	QNTU(IPW$DQR)		
PP50	Copy userid specification		QNTU(IPW$DQR)		
	Update pointers and process next keyword.....>	XK00		R1,R3	

IPW$$XR - VSE/POWER Execution Reader	
Label	Routine
XR00	Start New Partition Job
XR10	Writer-only Partition Support
XR22	Task Termination
XR32	Process Accounting Requests
XR42	Handle Output Requests
XQ00	Wait for User Request
XQ26	Emulate Channel Program
XQ82	End of User Job

Services Used	
Service	Macro
Task Management	IPW$ATT IPw$DET IPW$WFC IPw$WFQ IPW$WFS
Resource Management	IPw$RLR IPW$RSR
Storage Management	IPW$RLV IPW$RLW IPw$RSV IPW$RSW
Message Service	IPW$GAM IPW$NTY
Validation Service	IPW$VDA

Functions used	
Module	Macro
IPW$$FQ	IPw$FQS
IPW$$GD	IPW$GDR
IPW$$LU	IPW$ULP
IPW$$NQ	IPW$GQS
IPW$$PA/PF	IPw$PAR
IPW$$SL	IPW$GSL
IPW$$XJ	IPw$SXJ

Called by	
Module	Description
IPW$$I7	VSE/POWER Initialization
IPW$$PL	Physical List

Labels	Chart XR: 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 release'				
	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$DPD - Part. control block			R6	
	7: IPW$DCB - User command control block			R7	
	8: IPW$DCW - User channel command word			R8	
	9: XRCS - Base register			R9	
	10: IPW$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			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 XR: 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$RSW
	Save the real address of the queue space.		TCQA(IPW$DTC)	R0	
	Save the virtual address of the queue space.		TCQV(IPW$DTC)	R1	
	Reserve job header record space				IPW$RSV
	If request was not successful, branch to.....>	XR22			
	Save job header record address in partition control block		PDJH(IPW$DPD)		
XR10	Move job date and default name into queue record		QRDY (IPW$DQR) QRNM (IPW$DQR)		
	Update job number in the master record and insert it into queue record		MRNO(IPW$DQC) QRNO(IPW$DQR)		IPW$RSR IPW$RLR
	Insert default values in the queue space:				
	• Set default priority		QRPY(IPW$DQR)		
	• Set queue record identifier to C'C' to indicate console record		QRQI(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 VSE/POWER cancel code		QRCN(IPW$DQR)		

Labels	Chart XR: IPW$$XR - Execution Reader		Modified Data Fields	Reg. Usage	Calls
	Initialize Job Header Record:				
	Copy skeleton job header record into the space anchored on the partition control block			R0,R1 R2,R3	
	Insert additional defaults in the job header record:				
	• Copy date of job		NJHGETS		
	• Copy default job name		NJHGJNAM		
	• Copy job identifier		NJHGJID		
	• Set JES2 priority		NJHGPRIO (IPW$DNR)		
	If a job trailer record already has been prepared, branch to.....>	XR1C		R1	
	Branch and link to prepare a job trailer record.....>	XS10		RE	
	Set default record length to 80		TCRL(IPW$DTC)		
	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			
	Set up registers 0 and 4 for examining the remaining entries.			R0,R4	

Labels	Chart XR: IPW$$XR - Execution Reader		Modified Data Fields	Reg. Usage	Calls
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$WFS
	On return from task selection branch to..... >	XR11			
	Writer-only Partition Handler				
XR18	Move the channel command word to the TCB.		TCRW(IPW$DTC)		
	Link to the JECL statement analysis routine to handle the incoming statement.				IPW$SXJ
	Store the updated CCW address in the CCW.		CBCS(IPW$DCB)		
	Set the residual count to zero.		CBCT(IPW$DCB)		
	Set the necessary flags in the first communication byte.		CBC1(IPW$DCB)		
	Set the user partition dispatchable			R7	TREADY
	Check for a job in progress, and if so branch to..... >	XR11			
	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$DPD)		
	Reset the VSE/POWER control flags in the communication region to zero.				
	The reader device and all unit record partition assignments which relate to the devices in the PDB are unassigned.				IPW$ULP
	If a read request is still outstanding, the necessary flags are set in the CCB.		CBC1(IPW$DCB)		

Labels	Chart XR: IPW$$XR - Execution Reader		Modified Data Fields	Reg. Usage	Calls
XR28	The user partition is set dispatchable.				TREADY
XR30	The queue space is returned to the storage pool.				IPW$RLW
	The job header record and job trailer record space is released.				IPW$RLV
	Issue message 1Q33I to the operator.				IPW$GAM
	Release the PDB.				IPW$RLW
	Detach the Execution Read Task.				IPW$DET
	• The task is removed from the task selection list. • The TCB storage is released.				
	Process Accounting Requests				
XR32	If there is no queue space available yet, the request is ignored..... >	XR38			
	The total length of the execution account record (EAR) is calculated in register 1.			R1	
	If the request was initiated by an SVC 91 branch to..... >	XR34			
	SVC 90 Processing:				
	Add user account size plus 8 bytes for the block descriptor to the size of the EAR.			R1	
	If the EAR size exceeds the maximum, ignore the request and branch to... >	XR38			
	Reserve space for the EAR. Move the user account information to the EAR.				IPW$RSW
	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. Set up register 7 as a base register for the EAR.			R7	IPW$RSW

Labels	Chart XR: 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.		AE#L(AEDS)		
	• Move the length of the SIO table from the partition JA table.		AESL(AEDS)		
	• Move the DOS/VSE account record from the partition JA table.		AEJN(AEDS)		
	Write the execution account record.				IPW$PAR
	Release the work space for the EAR.				IPW$RLW
	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 dispatchable.				TREADY
	Reset the request entry in the PDB.		PDCB(IPW$DPS)		
	If this is a writer-only partition branch to..... >	XR11			
	Otherwise, branch to..... >	XQ00			

Labels	Chart XR: IPW$$XR - Execution Reader		Modified Data Fields	Reg. Usage	Calls
	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			
	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)		
	• Store next CCW address in CCB		CBCS(IPW$DCB)	R8	
	Set the user partition dispatchable.				TREADY
	If this is a writer-only partition. >	XR11			
	Otherwise, branch to..... >	XQ00			
XR44	Reserve storage for the TCB to be associated with the new task.				IPW$RSW
	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$RSW
	Set up register 1 as a base register for the new queue record.			R1	
	Initialize the new queue record:				
	• Copy values from the reader queue record.		QNBF(IPW$DQR) QNB2(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)		
	• Clear default disposition		QNFI(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 XR: 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 >	XR50			
	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$DTC)		
	Move option byte.		QNOP(IPW$DQR)		
XR50	Initialize the new task registers by storing them in the save area of the new TCB:				
	2: save area address		TNRD(IPW$DTC)	R2	
	4: IPW$DDE address		TNR4(IPW$DTC)	R4	
	6: IPW$DPD address		TNR6(IPW$DTC)	R6	
	Reserve storage for the data set header record (DSHR) for the new task				IPW$RSW
	If reserve request is not successful, branch to.....>	XQ06			
	Save DSHR address in TCB		TN3E(IPW$DTC)		
	Initialize the general section of the DSHR:		(IPW$DNR)		
	• Set up section length		NDHGLEN		
	• Set up general section ID		NDHGTYPE		
	• Set up modifier		NDHGMOD		
	If not print output, branch to.....>	XR53			
	• Set print node name		NDHGNODE/QNTN		
	• Set print remote name		NDHGRMT/QNTU		
	• Indicate print output		NDHGFLAG2		
	Branch to.....>	XR54			
XR52	• Set punch node name		NDHGNODE/QNTN		
	• Set punch remote name		NDHGRMT/QNTU		
	• Indicate punch output		NDHGFLAG2		
	If the target print/punch node name has the format 'Rnnn', with nnn numeric, then nnn is converted to binary and stored in the queue record		QNTJ(IPW$DQR)		

Labels	Chart XR: IPW$$XR - Execution Reader	Modified Data Fields	Reg. Usage	Calls
XR54	• Blank out some fields	NDHGPROC		
	• Set output class	NDHGCLAS		
	• Set variable record format	NDHGRCFM		
	• Set default record length to 512	NDhGLREC		
	• Set copy count	NDHGDSCT		
	• Blank out forms ID	NDHGFORM		
	Initialize the 3800 section of the DSHR:			
	• Set section length	NDHALEN		
	• Set 3800 section ID	NDhATYPE		
	• Set up modifier	NDHAMOD		
	Initialize the VSE/POWER section of DSHR:			
	• Set section length	NDHPLEN		
	• Set up VSE/POWER section ID	NDHPTYPE		
	• Set up modifier	NDHPMOD		
	• Set device type	NDHPIDEV		
	• Set priority	NDhPPRIO		
	• Set disposition	NDHPDISP		
	• Set user information	NDHPUSER		
	• Set number of separators	NDHPNSEP		
	• Set options	NDhPOPTN		
	• Set partition ID	NDHPPART		
	• Set target SYSID	NDhPSYID		
	• Set compaction table name	NDhPCOMP		
	Initialize the SETPRT parameter list of DShR:			
	• Indicate initialize printer	SPLFLAG1		
	• Set TRC=NO and DEBUG=NORM	SPLFLAG2		
	• Set flash count to 255	SPLFLSHC		
	• Set copy grouping to 1	SPLCOPYG		
	• Set copy group index to 1	SPLCINDX		
	• Set parameter list length	SPLLNGTH		
	Set default required flag in TCB	TCF3(IPW$DTC)		

Labels	Chart XR: IPW$$XR - Execution Reader		Modified Data Fields	Reg. Usage	Calls
	Attach the new writer task into the system.				IPW$ATT
	The owner address in the PDB is reset, so that control will be passed to the execution list task when a user request is received.		TLTC(IPW$DDE)	R1	
	If this is a writer-only partition. >	XR11			
	Otherwise, branch to..... >	XQ00			
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. If no matching entry is found, branch to..... >	XR76		R3	
XR74	Check if matching entry is a 3540. If so, branch to..... >	XR78			
XR76	Issue message 1Q881 INVALID 3540 UNIT				IPW$GAM
	Set cancel code and branch to..... >	XQ06	TCTT(IPW$DTC)		
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			

Labels	Chart XR: IPW$$XR - Execution Reader		Modified Data Fields	Reg. Usage	Calls
	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)		
	If the user accepts unrecoverable errors, branch to get the next command..... >	XQ72			
	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$DCB)		
	Issue message 1R30I.				IPW$GAM
	Set cancel code to X'1A' to indicate cancel due to I/O error.			R1	
XR84	Cancel user partition.				TREADY
	Branch to get the next command..... >	XQ74			
	Emulate Transfer in Channel:				
XR86	If CCW not on double word boundary. >	XR82			
	Load the address of the next CCW into register 8 and branch back..... >	XQ28		R8	
	Emulate SENSE Operation:				
XR88	If an invalid sense command was issued branch..... >	XR82			
	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	
XR89	If a NO-OP command, set residual count to zero and branch to..... >	XQ71	CBCT(IPW$DCB)		
	Wait for User Request:				
XQ00	Unpost the event control block in the TCB. Check the reader entry in the PDB for a user request.		TCEB+2(IPW$ETC)		
	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 XR: 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			
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 HOLD disposition is saved to preserve the input queue entry for later processing.		QRDI(IPW$LQR)		
	Otherwise, branch to..... >	XQ10			
XQ06	Set cancel code X'23', program request.			R1	
	If program cancel set in TCB, branch>	XQ06B			
	Set off dump options.				
	Set 'job-canceled-by-operator',X'24'			R1	
XQ06B	Cancel user partition.			R7	TREADY
XQ07	If SLI work space is present, it is released.				IPW$GSL
XQ08	Restore termination byte in TCB		TCTT(IPW$DTC)		
	Set the VSE/POWER cancel code in the queue record.		QRCN(IPW$DQR)		
	Indicate normal record in the TCB.		TCGP(IPW$DTC)		
	If a buffer is available branch to. >	XQ09			
	Reserve buffer space.				IPW$RSW
	Store the real and virtual buffer addresses in the TCB.		TCDA(IPW$DTC)		
	Point register 7 to the data buffer.			R7	
XQ09	Add 2 to the buffer address and store it as the previous record in the TCB.		TCPR(IPW$DTC)		
	Move a /& into the buffer.				
XQ10	Exit to task selection to wait for a user request.				IPW$WFC
	On return from task selection branch to..... >	XQ00			

Labels	Chart XR: 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$GQS
	If a record is obtained branch to... >	XQ18			
XQ15	If the task is in S or E state, branch ..... >	XR22			
	Indicate Q state.		POWFLQ1		
	Issue message 1Q34I to the operator.				IPW$GAM
XQ16	Exit to task selection to wait for a new entry to be added to the queue.				IPW$WFQ
	Check the termination indicator. If a PSTOP or PEND command has been issued branch to the task termination routine..... >	XR22			
	Get a new queue record.				IPW$GQS
	If no record is obtained, branch to >	XQ15			
	Reset message address.		TCMW(IPW$DTC)		
	Reset Q state indicator.		POWFLQ1		
	Set up register 5 as a base register for the queue record.			R5	
XQ18	Ready the task to obtain data records and to react to requests from the user program being executed in the related DOS/VSE partition:				
	• Move data seek address into disk request word.		TCDW(IPW$DTC)		
	• Save class for accounting.		TCGW+15(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)		
	If in the current queue entry the physical unit for 3540 data spooling is specified branch to..... >	XR70			
	Otherwise, branch to..... >	XQ00			

Labels	Chart XR: 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.			R8	
	Is device a 2501? No.....	> XQ27			
	Yes, Read ahead request?.....	> XQ26A			
	NO, Reset EOF switch and branch to.....	> XQ27			
XQ26A	EOF switch on? No.....	> XQ27			
	Yes, reset EOF switch and ignore request.....	> XQ71			
XQ27	For EXCP real, branch to handle invalid operation.....	> XR82			
XQ28	Validate CCW addresses.				IPW$VDA
	If not valid, branch to.....	> XR82			
	Classify Command:				
	If the IDAL flag or data chain flag is present in the channel command word, which feature is not supported by VSE/POWER, branch to.....	> XR82			
	Check the low-order four bits of the command code to determine which handling routine to branch to for handling the command:				
	• Invalid operation.....	> XR82			
	• Transfer in channel.....	> XR86			
	• Sense operation.....	> XR88			
	• Write operation.....	> XR82			
XQ30	• Control command.....	> XR89			
	• Read operation.....	> XQ46			
	Emulate READ Operation:				
XQ46	Validate data address and associated length provided in the user-supplied CCW. If not valid, branch to.....	> XR82			IPW$VDA
	Check whether the user program attempts to read past a /& statement. If not, branch to.....	> XQ48			
XQ47	Cancel code X'30' is set to indicate cancel due to reading past /& and a branch is made to cancel the task.....	> XR84		R1	
XQ48	If the previous command was read only, reset the read only switch and check current command.....	> XQ62			

Labels	Chart XR: IPW$$XR - Execution Reader		Modified Data Fields	Reg. Usage	Calls
XQ49	Check the pointer to the SLI work area in the PDB to determine whether the task is in SLI mode of operation. If not, branch to..... >	XQ52			
	Check whether the SLI routines require an additional record from the data file. If not, branch to..... >	XQ50			
	Read the next data record.				IPW$GDR
	If it is a data set header record, branch to..... >	XQ49			
	If new CCB address, process new request..... >	XQ00			
XQ50	Get the next record from the source statement library.				IPW$GSL
	If the flush job request has been returned, branch to..... >	XQ06			
	If new CCB address, process new request..... >	XQ00			
	If an update statement was processed, branch to..... >	XQ48			
	If the record is a data record, branch to..... >	XQ5B1			
	If the record does not begin with '* $', branch to..... >	XQ62			
XR60	Save request word.			R2,R3	
	If length of record not greater than 71, branch to..... >	XR62			
	Set length to 71.		TCRL+1(IPW$DTC)		
XR62	A call is made to IPW$$XJ to scan JECL record.				IPW$SXJ
	If the flush job request has been returned, branch to..... >	XQ06			
	If not a JECL record branch to..... >	XR64			
	Restore request word Branch to get next record..... >	XQ55		R2,R3	
XR64	Restore request word. Branch to..... >	XQ62		R2,R3	
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..... >	XQ5E			

Labels	Chart XR: IPW$$XR - Execution Reader		Modified Data Fields	Reg. Usage	Calls
XQ5A	Get the next data record from the data file.				IPW$GDR
XQ5B	Check if user CCB address changed. If yes, then branch to..... >	XQ00			
XQ5B1	If new record is not an internal control record, branch to..... >	XQ54			
	If new record is not a data set header record, branch to..... >	XQ5B2			
	Get maximum record length from data set header record and save it in the queue record and branch to..... >	XQ5A	QRRL(IPW$DQR)	R1	
XQ5B2	If it is a job trailer record, branch to..... >	XQ52			
	Branch and link to handle job header record..... >	XS00		R14	
	Branch and link to build skeleton job trailer record..... >	XS10		R14	
	Branch to get next record..... >	XQ5A			
XQ54	Check the new record to see whether it is a JECL statement. If not, branch to..... >	XQ56			
	Link to the JECL statement analysis routine (IPW$$XJ).				IPW$SXJ
	If the flush job request has been returned, branch to..... >	XQ06			
	If the statement is not JECL, branch to..... >	XQ56			
XQ55	Turn off read only switch		TCG2(IPW$DTC)		
	If change in CCB address, branch to process new request..... >	XQ00			
	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..... >	XQ5B			
	If not on, then turn switch on and read card..... >	XQ5A	TCG2(IPW$DTC)		
XQ5F	Turn read only switch off, bypass read..... >	XQ5B	TCG2(IPW$DTC)		
XQ56	Check if user CCB address changed. If yes, them branch to..... >	XQ00			
	Load the record address in register 0			R0	
	Load the record length in register 1.			R1	
	If not a 3540 request, branch .....>	XQ60			
	If it is a 3540 data record, branch .>	XQ58			
	Issue message 1Q89I and branch to cancel the job ..... >	XR82			IPW$GAM

Labels	Chart XR: 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, reset read only switch and branch to get the next record..... >	XQ46	TCG2(IPW$DTC)		
XQ62	Potential read only command, no.... >	XQ63			
	Is it a 2520B1, yes..... >	XQ63			
	Is it read only, no..... >	XQ63			
XQ62B	Set read only switch				
XQ63	Is programmer unit? Yes..... >	XQ66			
	Is it end of data? No..... >	XQ64			
	Is it 2501? No..... >	XQ65			
	Yes, set EOF switch and branch..... >	XQ65			
XQ64	Is it end of job? No..... >	XQ66			
	Set end of job switch in PIB.		PIBFLG2		
	Set EOF on reader.		CBC1(IPW$DCB)		
	Set unit exception.		CBSD(IPW$DCB)		
XQ66	Load the user record length into register 3.			R3	
	Check if user CCB address changed. If yes, them branch to..... >	XQ00			
	Load register 2 with maximum record length from queue record. If no length value available, assume length of 80.			R2	
XQ67	Set residual count to zero.		CBCT(IPW$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)		
	Move record to user.				
XQ69	If wrong length is posted, break I/O>	XQ72	CBSC(IPW$DCB)		
	Get Next Command:				

Labels	Chart XR: IPW$$XR - Execution Reader		Modified Data Fields	Reg. Usage	Calls
XQ71	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)		
	Set the user partition dispatchable.				TREADY
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.				IPW$RLW
	Clear the addresses in the TCB.		TCDA(IPW$DTC)		
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$DTC)		
	Branch to wait for the next user request..... >	XQ00			
	End of User Job:				
XQ82	If a skeleton job trailer record is present, it is filled up with the following values: • the job stop time • the totel number of lines • the total number of cards		NJTGSTOP NJTGALIN NJTGACRD (IPW$DNR)		
	If writer only partition, branch to..... >	XQ84			IPW$RLW
	Release data buffer.				
XQ84	Set up register 8 for shutting down any subordinate writer tasks. (R8 = address of first entry in PDB.)			R8	
	If not 'MT' and not JCL, branch to. >	XQ47			
	If partition starts as 'MT' (multi-tasking) branch to..... >	XQ93			
XQ86	Scan the entries in the PDB.				
	If DISP is not N, branch to..... >	XQ90			
	If the device entry contains a class code of 'N', the device class is reset to C'L' (for list) or C'P' (for punch).		TLCL(IPW$DDF)		

Labels	Chart XR14: IPW$$XR - Execution Reader		Modified Data Fields	Reg. Usage	Calls
XQ88	Release the ownership of the device, and branch to..... >	XQ92			IPW$ULP
XQ90	If a task has been started:				
	• Reset the ownership in the TCB.		TLTC(IPW$DDF)		
	• Set terminator type to C'S'.		TNTT(IPW$DTC)		
	• Post event control block.		TNEB+2(IPW$DTC)		
	Wait for signal from the list task.				IPW$WFC
XQ92	Load the address of the next entry into register 8.			R8	
	Check for the end of entries. If not, branch to..... >	XQ86			
	Check if this is a writer-only partition. If so, branch to..... >	XQ96			
	Check for a 3540 unit specification. If not, branch to..... >	XQ94			
	Release the 3540 device list entry:				
	• Zero the 3540 entry		PDER(IPW$DPD)		
	• Update the first entry address		PDPA(IPW$DPD)		
	• Update the number of entries		PDNE(IPW$DPD)		
XQ94	Delete the queue record from the queue.				IPW$DQS
	Release the queue record.				IPW$FQS
	If the job header record does not contain a user ID specification, branch to..... >	XQ95			
	Set up parameter register for notify message				
	R0: origin node name			R0	
	R2: user ID			R2	
	R4: own node name			R4	
	Issue notify message '1Q5DI'			R14	IPW$NTY
XQ95	Clear job header record pointer in the partition control block and release record space.		PDJH(IPW$DPD)		IPW$RLV
	Clear job trailer record pointer in the partition control block and release record space		PDJT(IPW$DPD)		IPW$RLV
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			

Labels	Chart XR14: IPW$$XR - Execution Reader		Modified Data Fields	Reg. Usage	Calls
	Job header record processing:				
XS00	Reserve space for job header record.			R1	IPW$RSV
	If request not successful, reset queue set and terminate task..... >	XR22			IPW$DQS
	Save record space address in partition control block.		PDJH(IPW$DPD)		
	Copy contents of the record addressed by the TCB into the new job header record space.				
	Insert the job name from the queue record.		NJHGJNAM		
	If job logging is requested write job log message '1Q47I'				IPW$GAM
	Return to caller..... >	R14			
	Job trailer record processing:				
XS10	Reserve space for job trailer record.			R1	IPW$RSV
	If request successful, branch to... >	XS15			
	If writer only partition, terminate task..... >	XR22			
	Reset queue set and terminate task..... >	XR22			IPW$DQS
XS15	Save record space address in partition control block		PDJT(IPW$DPD)		
	Set initial values of job trailer record:				
	• Set total length		NJTLEN		
	• Set general section length		NJTGLEN		
	• Set up section ID		NJTGTYPE		
	• Set up modifier		NJTGMOD		
	• Copy class from queue record		NJTGXCLS		
	• Set start time and date		NJTGSTRT		
	• Copy initial priority		NJTGIXPR		
	• Set actual priority = initial		NJTGAXPR		
	Return to caller..... >	R14			

IPW$$XW - VSE/POWER Execution Writer	
Label	Routine
X700	Start New Partition Job
XW60	End of User Job
XW58	Count-driver Segmentation
XW62	Program-driven Segmentation
XW68	Handle Break Condition
XX00	Wait for User Program Request
XX02	Emulate Channel Program

Services Used	
Service	Macro
Task Management	IPW$DET IPW$WFC
Resource Management	IPW$RLR IPW$RSR
Storage Management	IPW$RLV IPW$RLW IPW$RSV IPW$RSW IPW$UNV
Disk / Tape Service	IPW$CTT
Message Service	IPW$GAM
Validation Services	IPW$VDA

Functions used	
Module	Macro
IPW$$AQ	IPW$AQS
IPW$$FQ	IPW$FQS
IPW$$OT	IPW$OTP
IPW$$PD	IPW$PDR
IPW$$RQ	IPW$RQS
IPW$$XJ	IPW$SXJ

Called by	
Module	Description
IPW$$NU	Task Management

Labels	Chart XW: IPW$$XW - Execution Writer		Modified Data Fields	Reg. Usage	Calls
	This routine is initiated by the Execution Read Processor, which has acquired queue record and data set header record space and formed the queue and data set header record for this task.				
XWSD	The first 16 bytes constitute the section descriptor: 'XWCS release'				
	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 - VSE/POWER 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:				
	Setup second base register, using register 15 and save it in TCB.		TC15(IPW$DTC)	R15	
XV00	Reserve space for the data buffer in which the output records are to be formed.				IPW$RSW
	Store the data buffer addresses in the TCB.		TCDA(IPW$DTC)		
	The buffer control words (BCW) of the queue record and of the data set header record, which have been acquired by the execution reader task, are updated, so that the ownership references the proper execution writer task. (Note: This is necessary so that the storage can be released properly if abnormal termination of the task should occur.)			R1	IPW$UNV
	The jobnumber obtained from the reader queue entry is saved for accounting purposes. The first queue entry of each type has the same job number as the reader entry. Any subsequent entries receive unique job numbers.		QRJ#(IPW$DQR)	R1,R14	
	If first output entry, branch .....>	XV10			

Labels	Chart XW: IPW$$XW - Execution Writer		Modified Data Fields	Reg. Usage	Calls
XV06	Update job number so that all output may be easily manipulated by operator.				
	Reserve queue block.			R3	IPW$RSR
	Get next number for this queue entry.		QRNO(IPW$DQR)	R1	
	If number is >32767, set back to 1.		MRNO(IPW$DQR)	R1	
XV08	Release queue block.				IPW$RLR
XV10	If disposition not ='T', branch to. >	XV14			
	If a TBB already exists, branch to. >	XV12			
	Initialize TBB for output tape.			R2	IPW$OTP
	If the DISP has been changed to 'D'.>	XV14			
XV12	Call IPW$OTP to open tape for output.				IPW$OTP
	If the task is not forced to stop ..>	XV13			
	If the execution reader is at job end don't flush, branch .....>	XV66			
	Indicate flush and branch .....>	XV66	TNTT(IPW$DTC)		
XV13	Set number of tracks to zero.		QRNT(IPW$DQR)		
XV14	Reserve a queue record.			R2	IPW$RQS
	Set up record pointer.		TCBC(IPW$DTC)		
	Set up seek address.		TCDW(IPW$DTC)		
	If disposition not tape, branch to. >	XV16			
	If EOV not found, branch to..... >	XV16		R1	
	Inform operator of EOV and request new tape mount.				IPW$GAM
	Invoke open tape to close old tape and open next.				IPW$OTP
	Branch to..... >	XV14			

Labels	Chart XW: IPW$$XW - Execution Writer		Modified Data Fields	Reg. Usage	Calls
XV16	If not a 3800 printer, branch to... >	XV18		R2	
	Branch to get the system defaults ..>	XVA0		R14	
	Move FCB name and UCS name.		NDHGFCB,NDHGUCS (IPW$DNR)		
	If no MOD is specified, assume first character arrangement table.		SPLCMCHR (SPLIST)		
	Forms control and UCS buffer handling:				
XV20	If not a list task, branch to..... >	XV21			
	If special FCB not required branch .>	XV44			
	Check if FCB already in table .....>	XV75			
	If entry was found, branch to..... >	XV21			
	Link to set up load list..... >	XV76		R14	
	Link to load the phase..... >	XV74		R14	
	If an error occurred branch to..... >	XV42			
	Link to convert FCB to LTAB..... >	XV80		R14,R0	
	If an error occurred, branch to..... >	XV42			
	Link to update FCB table..... >	XV77		R14	
XV21	Save SYSID and partition ID in data set header record		NDHPSYID,NDHPPART (IPW$DNR)		
	If device is not a 3800 printer ....>	XV24			
	If transmission count is not zero set copy count equal transmission count.		QRNC(IPW$DQC)		
XV24	If copy count equals zero set it in queue record and data set header record to one.		QRNC(IPW$DQC) NDHGDSCT (IPW$DNR)		
	Write job header record .....>	XU79			
	Write data set header record .....>	XV78			
	Join mainline..... >	XX00			
	Error on FCB Load:				
XV42	Release the lock on the FCB table.			R3	IPW$RLR
	Issue message 1Q54I.				IPW$GAM
XV44	Copy default LTAB from the queue control block into the TCB..... >	XV90		R14	
	Clear out FCB name in the general section of the data set header record		NDHGFCB (IPW$DNR)		
	Set the FCB name to zero in the SETPRT parameter list of the data set header record		SPLFCB (SPLIST)		
	Write job and data set header record>	XV21			

Labels	Chart XW: 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..... >	XV61			
	Set disposition to 'D' and free queue set. Branch to..... >	XV66	QRDP(IPW$DQR)	R1	IPW$FQS
XV61	Backspace tape to overwrite any X'FF' record that has been written. Branch to..... >	XV66		R1,R14 R15	IPw$CTT
XV62	Set end of data and branch and link to write job trailer record..... >	XW79	TCGP(IPW$DTC)		
	Indicate end of segment in the queue record if not only segment.		QRSN(IPW$DQR)		
XV63	Check the disposition indicator to see whether the record has to be added to the reader queue. If not, branch to..... >	XV64			
	For reader entry, set the suffix number to zero.		QRSN(IPW$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$AQS
XV66	Return the data buffer space to the storage pool.			R1	IPw$RLW
	In case of tape processing close tape				IPW$OTP
XV68	Return the queue space, if present, to the storage pool.				IPW$RLW
	Reset the space addresses in the TCB.		TCQA(IPW$DTC)	R0,R1	
	Release the data set header record space			R1	IPW$RLV
	If task stops itself, branch to.... >	XV69			
	Return device list entry to the execution reader		TLTC(IPW$DDE)		
	Set the Execution read task to dispatchable.		TCSF(IPW$DTC)		
XV69	Detach the Execution writer task: • The task is removed from the task selection list. • The TCB storage is released.				IPw$DET

Labels	Chart XW: IPW$$XW - Execution Writer		Modified Data Fields	Reg. Usage	Calls
	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.		TCBL(IPW$DTC)		
	Set the option switch in the parameter list to X'00' to indicate loading of text.				
	Load the address of the parameter list into register 1.			R1	
	Load the phase by loading the entry point into register 0 and issuing an SVC 4.			R0	
	Load the load point address of the phase in register 3 and the directory list pointer in register 2.			R3 R2	
	If not a 3800 printer, branch back to caller.			R14	
	If the phase just loaded is not an FCB, take error exit.			R2	
	Get the FCB length out of the header and save it in the TCB.		TCBL(IPW$DTC)	R2	
	Bump with register 3 over the FCB header.			R3	
	Return to caller via link register R14.....>	4(R14)			

Labels	Chart XW: IPW$$XW - Execution Writer		Modified Data Fields	Reg. Usage	Calls
	Lock FCB Table and Search for Entry:				
XV75	Reserve the FCB table.			R3	IPW$RSR
XV75A	Check the table for the current FCB name. If the entry is not found return to caller with not found address..... >	4(R14)		R3,R1, R2	
XV75B	Move the LTAB from the FCB table into the TCB and store the page size also in TCB.		XWLC XWPC		
	Release FCB table.			R3	IPW$RLR
	Return to caller..... >	R14			
	Set-up Load Parameter List and Convert Generic Names to Actual Names:				
XV76	Move data address into the GENL statement.		LDLG		
	Move in FCB name.		LDPN		
	If the FCB name starts with '$$$$' replace by correct prefix: • FCB1 for 3800 • FCB2 for PRT1 • FCB3 for 3203-1 • FCB4 for 5203.				
	Return to user..... >	R14			
	Update FCB Table with New FCB Name and Converted LTAB:				
XV77	Set up addressability to the FCB table.			R1,R2	
	Look for a free entry (X'00') in the FCB table and if found branch to... >	XV77B			
XV77A	All the table entries are full so the first one loaded is moved out and all others moved down. The table has room for 9 entries.		FCBTAB		
XV77B	Store the FCB name, converted LTAB and actual page size used.		FCBNAM FCLTAB FCPSIZ	R3	
	Release the FCB table.			R3	IPW$RLR
	Return to caller..... >	R14			

Labels	Chart XW: IPW$$XW - Execution Writer		Modified Data Fields	Reg. Usage	Calls
	Write the data set header record:				
XV78	If the queue record has disposition 'I', return to caller.....>	R14			
	Store the data area address and record length in the TCB.		TCRV(IPW$DTC) TCRL(IPW$DTC)	R3	
	Move control code (X'FF') into the TCB.		TCCC(IPW$DTC)		
	Indicate data set header record.		TCG2(IPW$DTC)		
	Write the logical record.				IPW$PDR
	Reset data set header record indication.		TCG2(IPW$DTC)		
	Clear general purpose byte of the TCB		TCGP(IPW$DTC)		
	Reset some SETPRT parameter list flags.		SPLFLAG1,SPLFLAG2 (SPLIST)		
	Return to caller..... >	R14			
	Write job header record:				
XU79	If the queue record has not disp=I, and the target node and userid of the job header are equal to those of the queue record, branch to.....>	XV81			
	Otherwise a new job header is built:				
	• Reserve space for new JHR. If request not successful, branch.>	XV60			IPW$RSV
	• Copy old JHR and setup the new punch node and remote name from the queue record.		NJHGPUNN NJHGPUNR		
XU81	Set JHR address and length, control code X'FF', and JHR indicator in the TCB.		TCRV,TCRL TCCC,TCG2		
	Write the record.				IPW$PDR
	Reset JHR indicator and clear general purpose byte.		TCG2,TCGP		
	If the space of the just written JHR had been newly acquired, release it.				IPW$RLV
	Return to caller .....>	R14			

Labels	Chart XW: IPW$$XW - Execution Writer		Modified Data Fields	Reg. Usage	Calls
	Write job trailer record:				
XV79	Store the data area address and the record length in the TCB.		TCRV(IPW$DTC) TCRL(IPW$DTC)		
	Move control code (X'FF') into the TCB.		TCCC(IPW$DTC)		
	Indicate job trailer record.		TCG2(IPW$DTC)		
	Write the logical record.				IPW$PDR
	Reset job trailer record indication.		TCG2(IPW$DTC)		
	Clear general purpose byte.		TCGP(IPW$DTC)		
	Return to caller..... >	R14			
	Convert FCB to line Table:				
	The carriage control line table in the TCB is set according to the specified FCB image.				
	Register on entry:				
	R0 - Lengths of FCB image R3 - Points to 1st byte of FCB				
XV80	Clear out line table in TCB.		XWLC(IPW$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	
	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)		

Labels	Chart XW: IPW$$XW - Execution Writer		Modified Data Fields	Reg. Usage	Calls
XV94	All channel positions are now converted to positions relative to channel one.		XWLC(IPW$DTC)	R0,R1 R2,R3	
	Return to caller via register 14 with a displacement of 4..... >	4(R14)			
	Get System Defaults for 3800 Printer The system default information is obtained from the PUB2 area and moved in the SETPRT parameter list which is part of the data set header record.				
XVA0	Save callers registers R14-R9		SVRE(IPW$DSV)	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.				EXTRACT
	The needed parameter values are passed in the following registers: R0: length R1: pointer to device address R2: parameter list address R3: data area address			R0 R1 R2 R3	
	On return, test for successful completion. If not, issue message 1QB5I and return is made to the caller..... >	R14			IPW$GAM IPW$CNC
	Reestablish the second base register, which has been destroyed by the EXTRACT supervisor function. Now each field in the SETPRT parameter list is checked if specified. If not, the appropriate system default is inserted into the SETPRT parametet list, which is part of data set header record.			R15	
	If a FCB name is specified or no FCB default name is defined, branch >	XVA2			
	Save FCB name in the data set header record		SPLFCB(SPLIST)		

Labels	Chart XW: IPW$$XW - Execution Writer		Modified Data Fields	Reg. Usage	Calls
XVA2	If a character arrangement table is specified, branch to..... >	XVA4			
	Otherwise, set the default character arrangement table in the data set header record.		SPLCHAR1(SPLIST)		
XVA4	If a copy modification module is already specified, branch to..... >	XVA6			
	Otherwise, set default copy modification name in the data set header record.		SPLCMMOD(SPLIST) NDHTAB1(IPW$DNR)		
XVA6	If a flash id is already specified, branch to..... >	XVA8			
	Otherwise, copy the default flash id into the data set header record and the queue record.		SPLFLASH(SPLIST) QRFL(IPW$DQR) NDHAFLSH(IPW$DNR)		
XVA8	If a forms id is already specified, branch to..... >	XVB0			
	Otherwise, copy the default forms name into the data set header record and the queue record.		SPLFORMS(SPLIST) QRFI(IPW$DQR) NDHGFORM(IPW$DNR)		
XVB0	If BURST=Y|N is specified, branch to..... >	XVB4			
	Otherwise, take the default burst specification and save it also in the queue record.		SPLFLAG1(SPLIST) QRPS(IPW$DWR) NDHAFLAG1(IPW$DNR)		
XVB4	Clear out the portion of the logical data area which has been used as work area, obtaining the default settings out of the PUB2.			R2	
	Reload registers	R14-R9			
	Return to caller..... >	R14		R14	
	Validate 3800 FCB Image:				
XVC0	Since the subroutine uses more registers than are available, contents of registers 4 - 7 are saved in the own save area addressed by register 13.		SVR4(IPW$DSV)	R4-R7	
	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 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.				

Labels	Chart XW: IPW$$XW - Execution Writer		Modified Data Fields	Reg. Usage	Calls
XVC2	Locate the first printable line of the page.			R1,R3 R4,R5 R6	
	Clear out work registers 1 and 2.			R1,R2	
XCV4	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 and branch to >	XV90		R4-R7	
	Merge Current Printer Setup This subroutine merges the current printer setup contained in the SETPRT parameter list of the data set header record with the just obtained SETPRT-request. Registers on entry: R1 - SETPRT parameter list address R2 - SETPRT parameter list of data set header record R3 - data set header record address R14 - Return address Each field of the SETPRT parameter list is checked if specified. If so, the appropriate field in the data set header record (DSHR) is updated with the new value. In addition, if the new printer setup requires operator intervention, the segmentation-required flag is set.				
XVE0	If INIT=Y is specified in the SETPRT parameter list, set the appropriate flag in the DSHR.		SPLFLAG1(SPLIST)		
XVE2	If SEP=M is specified in the SETPRT parameter list, set the appropriate flag in the DSHR.		SPLFLAG1(SPLIST)		
XVE4	If SFP=0 is specified in the SETPRT parameter list, set the appropriate flag in the DSHR.		SPLFLAG1(SPLIST)		
XVE6	If FCB verification is specified, set the appropriate flag in the DSHR.		SPLFLAG2(SPLIST)		

Labels	Chart XW: IPW$$XW - Execution Writer		Modified Data Fields	Reg. Usage	Calls
XVE8	If DCHK=U (unblock data check) is specified in the SETPRT parameter list, set the appropriate flag in the DSHR.		SPLFLAG2(SPLIST)		
XVF0	If the DEBUG keyword is specified in the SETPRT parameter list, set the appropriate flag in the DSHR.		SPLFLAG2(SPLIST)		
XVF2	If BURST=Y|N is specified in the SETPRT parameter list, set the appropriate flag in the DSHR.		SPLFLAG1(SPLIST) NDHAFLG1(IPW$DNR) TCF3(IPW$DTC)		
XVF4	If TRC=Y|N is specified in the SETPRT parameter list, set the appropriate flag in the DSHR.		SPLFLAG1(SPLIST) NDHAFLG1(IPW$DNR)		
XVG0	If an FCB name is specified in the SETPRT parameter list, save FCB name in DSHR.		SPLFCB(SPLIST) NDHGFCB(IPW$DNR)		
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 DSHR and set the segmentation-required flag.		SPLCOPYG(SPLIST) NDHACPYG(IPW$DNR) TCF3(IPW$DTC)		
	Calculate the number of transmissions needed.		QRTC(IPW$DQR)	R0,R2 R3	
	Branch to continue..... >	XVG6			
XVG5	If INIT=Y was specified, set the first copy group value and the transmission count to one. Set segmentation-required flag.		SPLCOPYG(SPLIST) TCF3(IPW$DTC)		
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 DSHR.		SPLCINDX(SPLIST)		
	Branch to..... >	XVG8			
XVG7	If INIT=Y was not specified, branch to..... >	XVH0			
	Set CINDX value to one.		SPLCINDX (IPW$DTE)		
XVG8	Set the segmentation-required flag.		TCF3(IPW$DTC)		

Labels	Chart XW: IPW$$XW - Execution Writer		Modified Data Fields	Reg. Usage	Calls
XVH0	If a forms value different from the previous one is specified, update the DSHR and set the segmentation-required flag.		SPLFORMS(SPLIST) TCF3(IPW$DTC) NDHGFORM(IPW$DNR)		
XVH4	If FLASH was specified, update flash id and flash count in the DSHR. If INIT=Y was specified but no flash id, reset flash id and count in DSHR.		SPLFLASH SPLFLSHC(SPLIST) SPLFLASH SPLFLSH(SPLIST)		
XVI0	If CHARS has been specified in the SETPRT request, the CHARS values are updated in the DSHR.		SPLCHARS NDHTAB1,2,3,4 (IPW$DNR)		
XVI2	If MODIFY was specified, update appropriate fields in the DSHR.		SPLCPMOD SPLCMCHR(SPLIST)		
	Return to caller via register 14... >	R14			
	Handle SETPRT Request:				
XW00	Check if the device is a 3800 printer. If not, branch to..... >	XW88			
	Address the SETPRT parameter list using register 1.			R1	
	The DSHR containing the current printer setup is addressed by register 3.			R3	
	Check each field of the SETPRT parameter list. If specified, the appropriate field in the DSHR is updated with the new value by branching to..... >	XVF0		R14	
XW06	If no segmentation has been forced, branch to..... >	XW20			
	(Note: Segmentation is done when the new SETPRT request requires a different printer setup for BURST, FORMS, FLASH, COPYG, and CINDX.)				
	Link to the segmentation routine... >	XW40		R1	
XW12	Update the queue record with the values obtained from the SETPRT request: • Forms-id • Flash-id • Copy groupings • Paper thread request		QRFI(IPW$DQR) QRFL(IPW$DQR) QRCG(IPW$DQR) QRPS(IPW$DQR)		
XW15	Check if the CINDX value is > 1. If so, the copy count is set to one, assuming that the user will manage the transmission himself.		QRNC(IPW$DQR) QRCI(IPW$DQR)		

Labels	Chart XW: 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$DQR)		
XW18	If not tape spooling branch to..... >	XW20			
	Reserve a queue record for tape.			R1,R14 R15	IPW$RQS
	If a different FCB name is specified in the SETPRT parameter list it must be verified.				
XW20	If no new FCB is required, branch to..... >	XW24			
XW22	Save the new FCB name.		NDHGFCP(IPW$DNR)		
	Link to copy the default LTAB..... >	XV90		R14	
	If no new FCB or a special FCB for SETPRT is requested, branch to..... >	XW24			
	Check to see if FCB is already in the table..... >	XV75		R14	
	If FCB found in table branch to.... >	XW24			
	Link to set-up load list..... >	XV76		R14	
	Link to load phase..... >	XV74		R14	
	If an error occurs on load, branch to..... >	XW26			
	Validate the FCB image..... >	XV80		R14	
	Store the new name in FCB table.... >	XV77		R14	
XW24	If segmentation not required, branch>	XW25			
	Write the job header record.....>	XV79			
	Turn off segmentation flag.		TCF3(IPW$DTC)		
XW25	Link to write the data set header record..... >	XV78		R14	
	Branch to get next user CCW..... >	XX76			
XW26	Unlock the FCB table.				IPW$RLR
	Issue message: '1Q54I FCB/UCS ERROR.'				IPW$GAM
	Clear out FCB name in the DSHR (SETPRT parameter list).		NDHGFCB(IPW$DNR) SPLFCB(SPLIST)	R2	
	Write out control record(s) .....>	XW24	TEFCBN(IPW$DTE)		

Labels	Chart XW: IPW$$XW - Execution Writer		Modified Data Fields	Reg. Usage	Calls
	Handle Q-SETPRT Request:				
XW36	Move the proper system/programmer logical unit number out of the CCB into the SETPRT parameter list of DSHR.		SPLLUSYS(SPLIST)		
	Get the address of the data field in register 1 and the length in register 2.			R1,R2	
	Move the current SETPRT parameter list into that area.				
	Branch to get next operand..... >	XX76			
	Output Segmentation:				
	This routine is invoked when program segmentation is requested either by LFCB macro or a SETPRT request.				
XW40	If there is already some output, branch to..... >	XW42			
	If it is not tape spooling, indicate reset of I/O area and pass request to data management, return to caller ..>	R1	TCRL(IPW$DTC)		IPW$PDR
	Backspace file to overwrite any X'FF' record and go to reinitialize the queue record..... >	XW44			IPW$CTT
XW42	Set end of data indicator and write job trailer record..... >	XV79	TCGP(IPW$DTC)		
	Add the queue set to the file. If we have tape spooling, the reserve of the queue record is delayed..... >	XW44		R11	IPW$AQS
	Reserve a new queue record.			R11	IPW$RQS
XW44	Initialize all count fields within the queue record.		QRNA(IPW$DQR) QRNR(IPW$DQR)		
XW46	The suffix number is set to zero.		QRSN(IPW$DQR)		
	The DMB is reserved, the current master job number is retrieved and updated by one, then the DMB is released.		MRNO(IPW$DQC) QRNO(IPW$DQR)	R3 R2	IPW$RSR IPW$RLR
	Return to caller..... >	R1			
	Count-driven Segmentation:				
XW58	Save current request word.			R0,R1	
	Set end of data and write job trailer record.....>	XV79	TCGP (IPW$DTC)	R2	
	Restore request word.		TCRW(IPW$DTC)	R0,R1	
	Add the current queue set to the class chain.			R2	IPW$AQS
	If intermediate storage is not magnetic tape, branch to..... >	XW59		R0	

Labels	Chart XW: IPW$$XW - Execution Writer		Modified Data Fields	Reg. Usage	Calls
XW5T	Reset segmentation.		QRBS(IPW$DQR)		
	Indicate close of tape and link to close tape function.		TCTF(IPW$DTC)	R2	IPW$OTP
	If stop is set, branch to..... >	XV60			
XW59	Reserve a new queue record.				IPW$RQS
	If disp=T and EOV found, branch to. >	XW5T			
XW5A	Update the request word in the TCB.		TCDW(IPW$DTC)		
	Reset the record counts in the queue record to zero.		QRNA(IPW$DQR) QRNR(IPW$DQR)		
	Issue message 1Q53I.				IPW$GAM
	Write out the job header record.... >	XU79		R14	
	and the data set header record..... >	XV78			
	Branch to..... >	XX38			
	Program-driven Segmentation:				
XW62	Link to segmentation routine..... >	XW40		R1	
XW66	Set the forms identifier to blanks.		QRFI(IPW$DQR)		
	Validate data area address. If the address is invalid branch to..... >	XW88		R0	IPW$VDA
	Save the new FCB name in the data set header record.		NDHGFCB(IPW$DNR)	R3	
	If a forms ID is present the new forms ID is moved to the queue record and the data set header record				
	If not, branch to..... >	XW67	NDHGFORM(IPW$DNR)		
XW67	Is it tape spooling?				
	If not, branch to..... >	XW67B			
	Reserve Queue record.			R11	IPW$RQS
XW67B	Force write of job header record.		TCF3(IPW$DTC)		
	Branch to check the FCB table ..... >	XW23			

Labels	Chart XW: 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$DTC)		
	Release the data buffer space.				IPW$RLW
	If no break condition, 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$WFC
	Check for a data break condition. If so, ignore it and branch to..... >	XW70			
	Recover from Break Condition				
XW74	Reserve new buffer space.				IPW$RSW
	Store the new buffer addresses in the TCB.		TCDA(IPW$DTC)		
	Branch to process the next user request..... >	XX00			
	NOP Creation Subroutine:				
XW76	This subroutine creates a NOP record to pass the end of data or end of block condition for the queue record.				
	• Set data address to blanks		TCCC(IPW$DTC)		
	• Set length to one		TCRL(IPW$DTC)		
	• Set NOP command code		TCCC(IPW$DTC)		
	Pass the record to the data file.			R0	IPW$PDR
	Return to caller via link register 2.			R2	
	Wait for User Program Request:				
XX00	Check the termination indicator in the TCB for a data break condition. If so branch to handle the break condition..... >	XW68			
	Check the termination indicator for a stop condition. If so, branch to.. >	XW60			
	Unpost the event control block in the TCB.		TCEB+2(IPW$DTC)		
	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			
	Exit to task selection.				IPW$WFC
	Branch to check for a user request..... >	XX00			

Labels	Chart XW: IPW$$XW - Execution Writer		Modified Data Fields	Reg. Usage	Calls
	Emulate Channel Program:				
XX02	Set up register 8 as a base register for the channel command word.			R8	
	Check for EXCP real, which VSE/POWER does not support when issued from a virtual partition.				
	If so, branch to..... >	XW88			
	Classify Command:				
XX04	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			IPW$VDA
XX03	If the IDAL flag or data chain flag is present in the channel command word, which feature is not supported by VSE/POWER, branch to..... >	XW88			
	Check the low-order four bits of the command code to determine which handling routine to branch to for handling the command:				
	• Invalid operation..... >	XW88			
	• Tranfer in channel..... >	XW90			
	• Sense operation..... >	XX12			
	• Read operation..... >	XX16			
	• Write operation..... >	XX18			
	• Control command..... >	XX42			
	Emulate Sense Operation:				
XX12	If an invalid sense command was issued branch to..... >	XW88			
XX13	Validate data area address. If the address is invalid, branch to..... >	XW88			IPW$VDA
	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			

Labels	Chart XW: IPW$$XW - Execution Writer		Modified Data Fields	Reg. Usage	Calls
	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			
	If disposition 'I', branch to get next command..... >	XX76			
	Validate data area address. If the address is invalid, branch to..... >	XW88		R2	IPW$VDA
	Move command code and dummy I/O area address into record control word of TCB		TCRW(IPW$DTC)	R1	
	Set record length to one and indicate card motion.		TCGP(IPW$DTC)		
	Continue main line..... >	XX32			
	Emulate Write Operation:				
XX18	Validate data area address. If the address is invalid, branch to..... >	XW88			IPW$VDA
	If the command code represents a SFTPRT 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			
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			

Labels	Chart XW: IPW$$XW - Execution Writer		Modified Data Fields	Reg. Usage	Calls
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$PDR
	Branch and link to the line count subroutine for updating the line counter ..... >	XW72		R14	
	If the CCB address has changed, branch to..... >	XX00			
	Set command code to X'8B' to indicate skip to channel one.		TCCC(IPW$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 5425 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			
	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	

Labels	Chart XW: IPW$$XW - Execution Writer		Modified Data Fields	Reg. Usage	Calls
XX34	Check for skip-to-channel-one command. If not, ignore segmentation and branch to..... >	XX38			
	Load current page count.			R1	
XX36	If segmentation limit has been reached, branch to segmentation routine..... >	XW58			
	Store current record length if it is greater than the value already saved.		QRRL(IPW$DQR)		
XX38	Pass record to data file.			R0	IPW$PDR
	If the CCB address has changed .....>	XX00			
	If X'63' command then indicate end of block and write NO-OP..... >	XX64		R2	
XX39	If the CCB address has changed, branch to..... >	XX00			
	If the disposition indicates tape processing, and the reflective spot on the tape has been hit, force segmentation for next request.		QRBS(IPW$DQR)		
XX40	If the previous command was a control command, do not update the data transfer command and branch to..... >	XX68			
	Branch to count routine..... >	XX64			
	Emulate Control Command:				
XX42	If the command is not a NOP, then branch to..... >	XX43			
	Validate data address.			R2	IPW$VDA
	If invalid, ignore command..... >	XX76			
	Is 1st 4 characters of data area '* $$'?				
	If not, branch to..... >	XX76			
	If so, set up request word.		TCRW(IPW$DTC)		
XX4A	Scan JECL record.			R2	IPW$SXJ
	If JECL statement, branch to..... >	XX4B			
	If the user accepts unrecoverable I/O error, post it for him..... >	XW88			
XX4B	If this writer lost control, set stop code 'E'. Continue mainline..... >	XX76	TCTT(IPW$DTC)		
XX43	If the command is a printer command, branch to the printer control routine..... >	XX54			

Labels	Chart XW: 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>	XX46			
	If the command is a load print buffer command, treat it as a write command>	XX18			
XX46	Form the record control word in the TCB:				
	• Move the user CCW.		TCRW(IPW$DTC)		
	• If the command is a card motion command and the device is a 3525P, 3525RP, or a 2540P, set the general purpose byte to X'01' to indicate data record.		TCGP(IPW$DTC)		
XX50	• Otherwise, set the general purpose byte to X'00'.		TCGP(IPW$DTC)		
XX52	Handle punch control command .....>	XX30		R1	
	Emulate Printer Control Operations				
XX54	If the command code represents an LFCB operation code branch to..... >	XW62			
	Update line conter .....>	XX88			
	If the command code is not a skip or space branch to..... >	XW78			
XX56	Form the record control word in the TCB:				
	• Move the command code.		TCRW(IPW$DTC)		
	If the command is a 'Skip to Channel one' or not an immediate command,...>	XX57			
	If command is not 'Advance to end of sheet', branch .....>	XX56A			
	If printer is 3203-1, branch .....>	XX55			
	Else emulate invalid operation .....>	XW88			
XX56A	If previous command was a 'Print No Space', branch to .....>	XX58			
	Else branch .....>	XX57			
XX55	Move command code to TCB.		TCRW(IPW$DTC)		
XX57	Store address of dummy I/O area into the TCB, set record length to 1 ....>	XX32	TCRW(IPW$DTC) TCGP(IPW$DTC)	R1	
	Combine the previous Print No Space command with the current immediate command. Branch to .....>	XX76			

Labels	Chart XW: IPW$$XW - Execution Writer		Modified Data Fields	Reg. Usage	Calls
	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$GAM
	If the CCB address has changed, branch to..... >	XX00			
XX70	Check the record type for a punch record. If so, bypass the page counting routine and branch to..... >	XX76			
	Check the control character in the record just written. If it does not represent a skip to channel one, branch to get the next user CCW.... >	XX76			
	Increment the number of pages spooled in the PDB.		PD#P(IPW$DPD)	R2	
	Increment the number of pages in the queue record.		QRNP(IPW$DQR)	R1	
	Get Next Command:				
XX76	Load the address of the next channel command into register 8.			R8	
	Check the command for command chaining. If not, branch to..... >	XX78			
	If channel 9 was posted, branch to break the chain..... >	XX78			
	If channel 12 was not posted, branch to process the new command..... >	XX04			
XX78	Indicate no cancel by setting register 0 to zero.			R0	
XX80	Indicate completion of the channel program to the user: • Store the updated CCW address. • Set the residual count to zero. • Set the necessary flags in the communication byte.		CBCS(IPW$DCB) CBCT(IPW$DCB) CBC1(IPW$DCB)	R8	

Labels	Chart XW: IPW$$XW - Execution Writer		Modified Data Fields	Reg. Usage	Calls
XX84	Set the user partition dispatchable.				TREADY
	If a cancel condition has occurred (register 0 = nonzero), indicate cancel due to I/O error and cancel the partition.				TREADY
XX86	If the CCB address has changed,.....>	XX00			
	Reset the task entry in the TCB.		TLCB(IPW$DTC)	R7,R8	
	Wait for next user request.....>	XX00			
	Carriage Control Emulation:				
XX88	Get the command code, which represents either a skip or a space code in register 1.			R1	
	If the command code represents a Print No Space, return to caller via register 14 with a diplacement of 4.			R14	
	If the command code is not a Skip or Space code, return to caller via register 14.			R1	
				R14	
XX90	If the command code represents a skip code, branch to..... >	XX94		R1,R2	
	Calculate the new value of the line count in register 2.			R2	
	If not carriage channel 9 processing>	XX91			
	If necessary post channel 9 overflow.		CBC2(IPW$DCB)		
	Set unit check and branch to check if the page is filled .....>	XX92	CBSD(IPW$DCB)		
XX91	Set master record addressibility.			R3	
	If channel 12 is not in the LTAB specified, branch to..... >	XX92			
	If the current line count is lower than LTAB channel 12, then branch to>	XX92			
	If option MULT12 in the master record is on, branch to..... >	XX91A			
	If channel 12 already passed, branch to..... >	XX92			
XX91A	Post channel 12 overflow.		CBSD(IPW$DCB)		
XX92	If the page is not filled, branch ..>	XX96	XWPC(IPW$DTC)		
	Decrement count by page size.				
XX93	A dummy record is written to the data file to indicate page filled:				
	* Set dummy CCW code and length 1.		TCCC,TCGP		
	* Set address of dummy data area.		TCRV(IPW$DTC)		
	* Put data record				IPW$PDR
	Increment page count by 1.				
XX94	Calculate the new skip value in R2.			R2	
XX95	Store it in the TCB.		XWLC(IPW$DTC)	R2	
	Return to caller.	4(R14)		R14	

Labels	Chart XW: IPW$$XW - Execution Writer		Modified Data Fields	Reg. Usage	Calls
	Handle Special Printer Commands:				
XW78	Set up register 2 and check the command code against the entries in the printer command table.				
	If a match is found, branch .....>	XW82			
	If the command code is not in the table, it represents an immediate command, branch to..... >	XX56			
XW82	If specified printer is correct ....>	XW85			
	If the command is invalid branch to >	XW88			
XW85	If the command should be ignored ...>	XX76			
	If not a 3800 printer, branch to... >	XW86			
	If unblock data check command code, set the unblock data check flag in data set header record and branch ..>	XW86	SPLFLAG1(SPLIST		
XW8A	If block data check command code, turn off the unblock data check flag in the data set header record and branch to..... >	XW86	SPLFLAG1(SPLIST)		
XW8B	If not initialize printer command code, branch to ..... >	XW8C			
	Clear out current printer setup in data set header record.		SPLFLAG1(SPLIST) SPLLUSYS(SPLIST)		
	Set initialize printer flag.		SPLFLAG1(SPLIST)		
	Get default line table ..... >	XV90			
	Branch to..... >	XW24			
XW8C	If a load FCB command code, set the FCB unknown flag in the data set header record.		SPLFCB(SPLIST)		
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			IPW$VDA
	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 > Branch to the write routine..... >	XW88 XX21			

Labels	Chart XW: IPW$$XW - Execution Writer		Modified Data Fields	Reg. Usage	Calls
	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$GAM
	Set up the cancel code X'1A' in register 0 and branch to..... >	XX80		R0	
	Emulate Transfer in Channel:				
XW90	Validate data area address. If the address is invalid, branch to..... >	XW88		R2	IPW$VDA
	If the address of the next CCW is not on doubleword boundary, branch to.. >	XW88		R1	
	Increment TIC-command counter by 1.				
	If more than 255 TIC-commands have been specified in a single channel program, branch to..... >	XW88	TLCB(IPW$DDE)	R1	
	Save new TIC-command counter.		TLCB(IPW$DDE)		
	Load the address of the next CCW into register 8 and branch back..... >	XX04		R8	

READER'S
COMMENT
FORM

VSE/POWER
Program Logic Manual Part 2
Order No. LY12-5028-2

This manual is part of a library that serves as a reference source for system analysts, programmers, and operators of IBM systems. This form may be used to communicate your views about this publication. They will be sent to the author's department for whatever review and action, if any, is deemed appropriate. Comments may be written in your own language; use of English is not required.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatever. You may, of course, continue to use the information you supply.

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

Possible topics for comment are:

Clarity Accuracy Completeness Organization Coding Retrieval Legibility

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

_____

_____

_____

What is your occupation?_____

Number of latest Newsletter associated with this publication:_____

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

LY12-5028-2

--Cut or Fold Along Line--

Reader's Comment Form

Fold and tape | **Please Do Not Staple** | Fold and tape

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 40 ARMONK, N.Y.

POSTAGE WILL BE PAID BY ADDRESSEE:

International Business Machines Corporation
Department 812 BP
1133 Westchester Avenue
White Plains, New York 10604

Fold and tape | **Please Do Not Staple** | Fold and tape

IBM®

VSE/POWER Program Logic Manual Part 2 (File No. S370/4300-36) Printed in U.S.A. LY12-5028-2