IBM z/VSE/
VSE Central Functions

VSE/POWER
Diagnosis Reference

Version 7 Release 1

SC33-6322-07

IBM z/VSE/
VSE Central Functions

VSE/POWER
Diagnosis Reference

Version 7 Release 1

SC33-6322-07

Note!

Before using this information and the product it supports, be sure to read the general information under “Notices” on page xvii.

Seventh Edition (July 2006)

This edition applies to Version 7 Release 1 of IBM Virtual Storage Extended/POWER, which is part of VSE Central Function, Program Number 5609-ZVS, and to all subsequent releases until otherwise indicated in new editions or Technical Newsletters.

Order publications through your IBM representative or the IBM branch office serving your locality. Publications are not stocked at the addresses given below.

A form for readers' comments is provided at the back of this publication. If the form has been removed, address your comments to:

IBM Corporation
Attn: Dept ECJ - BP/003D
6300 Diagonal Highway
Boulder, CO 80301,
U.S.A.

or to:

IBM Deutschland Entwicklung GmbH
Department 3252
Schoenaicher Strasse 220
D-71032 Boeblingen
Federal Republic of Germany

When you send information to IBM, you grant IBM a non-exclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

Contents

Figures

Notices

References in this publication to IBM* products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any of the intellectual property rights of IBM may be used instead of the IBM product, program, or service. The evaluation and verification of operation in conjunction with other products, except those expressly designated by IBM, are the responsibility of the user.

This publication is intended primarily for use by IBM personnel responsible for program service. The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. It is not intended as a description of a programming interface. The use of this information is a customer reponsibility. Service for errors, ommissions, accuracy, or completeness will not be provided.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785, U.S.A.

Programming Interface Information

This publication is intended to help the customer to do diagnosis of VSE/ESA. This publication documents information that is Diagnosis, Modification, or Tuning Information provided by VSE/ESA.

Warning: Do not use this Diagnosis, Modification, or Tuning Information as a programming interface.

Trademarks and Service Marks

The following terms, used in this publication, are trademarks or service marks of the IBM Corporation in the United States and/or other countries.

```
ACF/VTAM
Advanced Function Printing
AFP
AS/400
CICS
CICS/VSE
Enterprise Systems Architecture/370
Enterprise Systems Architecture/390
ES/9000
ESA/390
IBM
OS/390
Print Services Facility
System/370
VM/ESA
z/VSE
VTAM
```

Preface

This manual, contains:

- **Chapter 1**: Gives an overview of VSE/POWER, states requirements for operation, and lists the devices supported by VSE/POWER.
- **Chapter 2**: Describes the method of operation, including linkage and register conventions.
- **Chapter 3**: Outlines the logical structure of VSE/POWER; it explains the internal operations and shows the relationships between tasks and routines.
- **Chapter 4**: Lists program identifiers. This allows you to establish the relationship between phases, modules, and control sections. This part also lists the VSE/POWER messages and the relating modules.
- **Chapter 5**: Describes the layout of the VSE/POWER partition, account records, control blocks, and the work areas required by VSE/POWER.
- **Chapter 6**: Gives debugging hints, and shows how you can get information from a dump of the VSE/POWER partition.

At the back of this manual, you find:

- **VSE/POWER Internal Macros**.
- **Appendixes**: They expand on the information given in the above sections.
- **List of Abbreviations**.
- **Bibliography**: Lists manuals you may want to consult.
- **Glossary**: Explains some of the terminology used in this manual.
- **Index**.

Chapter 1. Introduction

This section contains an overview of the Virtual Storage Extended/Priority Output Writers, Execution Processors and Input Readers (VSE/POWER) Program Product. It is organized as follows:

- *Purposes of VSE/POWER.* A general description of VSE/POWER and the way its major functions are performed under VSE/AF.
- *Communication with VSE/POWER.* A summary of the VSE/POWER Operator Commands and the Job Entry Control Language, which allow the user to control VSE/POWER operations. The format of the messages issued by VSE/POWER is also explained.
- *Environmental Requirements.* The programming requirements for the various functions of VSE/POWER, and the basic organization of the VSE/POWER partition with its storage requirements. The machines and devices which are supported by VSE/POWER are listed under "Hardware Support".

Purposes of VSE/POWER

VSE/POWER performs automatic spooling and priority scheduling under the control of the VSE/AF supervisor. VSE/POWER occupies a virtual partition in which it is initiated and can service from one to eleven batch partitions (other than the VSE/POWER partition) of a lower or sometimes a higher dispatching priority. Coexisting with the spooled static partitions, VSE/POWER services dynamic partitions. Input to supported partitions is first spooled onto intermediate disk storage. When the supported partition commences execution, I/O requests to reader devices are intercepted and satisfied from intermediate storage via I/O data areas in the VSE/POWER partition. Output requests to list and punch devices are also intercepted, with the output being stored in output data areas of the VSE/POWER partition and later transferred to disk or tape. Printing and punching of the output from disk or tape is carried out when requested by the operator. Under the control of VSE/POWER, programs may be executed in either real or virtual mode.
The optional Shared Spooling function permits the sharing of the VSE/POWER files that contain the spooled input and output among two or more VSE/AF systems running in the same processor or different processors.

The use of the optional networking function, referred to in other parts of this manual as PNET, allows VSE/POWER to fully participate within networks consisting of other VSE/POWER nodes, JES2/JES3 NJE nodes or RSCS nodes.
Jobs, Output (list, punch), operator commands and messages can be transmitted from one computer system to another.
The methods of communication used are binary synchronous communication (BSC) lines, channel-to-channel adapters (CTCA), synchronous data link control (SDLC) lines, and Internet (Transmission Control Protocol/Internet Protocol: TCP/IP).

Three major operations are performed under VSE/POWER control:

Read User job information is read from a reader device (card, diskette, or tape) and spooled to intermediate storage (DASD). The PUTSPOOL macro interface can be optionally used to submit a job stream from the user's buffer area to intermediate storage.

The input may also come optionally from a remote terminal supported by the VSE/POWER Remote Job Entry, or from a network supported by the VSE/POWER Networking function.

The job is executed under the control of the VSE/POWER execution processor to meet user program requests.

List List output generated by the user program is spooled to intermediate storage (DASD or magnetic tape) before being transferred to a list device, normally a line printer.

The GETSPOOL macro interface can be optionally used to request retrieval of printer output.

The list output may optionally be returned to a remote printer supported by the VSE/POWER Remote Job Entry, or to another node within a network supported by VSE/POWER Networking.

Punch Punch output generated by the user program is spooled to intermediate storage (DASD or magnetic tape) before being transferred to a punch device, normally a card punch.

The GETSPOOL macro can be used to request retrieval of punched output.

The punch output may optionally be returned to a remote punch supported by the VSE/POWER Remote Job Entry, or to another node within a network supported by VSE/POWER Networking.

Additionally, VSE/POWER provides a programmable interface, referred to in other parts of the manual as Spool Access Support (SAS), which allows accessing the VSE/POWER spool files from any application running in another partition either controlled or not controlled by VSE/POWER. It offers services that allow two-way communication between VSE/POWER and a user using the cross partition communication facility (XPCC) of VSE/AF. A "User" can be defined as any application, task, transaction, or interactive user who is active in another partition. The support enables a user to:

- Submit a job to the VSE/POWER RDR queue for later execution in a VSE/POWER controlled partition or to the XMT queue for transmission to another node in the network.
- Spool list or punch data to the output queues (LST, PUN or XMT).
- Retrieve data from the RDR, LST or PUN queue.
- Manipulate the local or remote queues.
- Route messages to the central operator, any remote operator, any other interactive user either on the local or remote system or any remote RJE terminal operator.
- Pass other commands, such as PDISPLAY T, to VSE/POWER for processing.
- Return job event messages

Intermediate storage therefore contains user input and output data spooled to and from it under the control of VSE/POWER.

The user can generate several different versions, depending on the parameters specified in the POWER macro, and optional PLINE, PRMT, PNODE and PCPTAB macros.

The partition in which VSE/POWER is initiated is referred to as the VSE/POWER partition.

VSE/POWER Private Subtasks

To perform the required operations concurrently, VSE/POWER is structured into a series of asynchronously executed tasks. These tasks are:

- VSE/POWER partition VSE/AF maintask
- VSE/POWER partition VSE/AF subtasks

In addition, VSE/POWER has its own

- VSE/POWER partition private subtasks

VSE/POWER private subtasking task support is provided within the VSE/POWER system and does not presuppose Multitasking Support within the VSE/AF supervisor.

The VSE/POWER private subtasks are as follows:

INITIATOR TASK. Takes over from the VSE/POWER maintask at startup time and completes VSE/POWER initiation. For details refer to “Initialization of VSE/POWER” on page 39.

TERMINATOR TASK. Created at initialization time, and waits for posting of either

- offline formatting of data file extents, or
- final deletion of entries in the DELetion queue, or
- termination processing due to the PEND command.

For details refer to “Termination of VSE/POWER” on page 54.

COMMAND TASK. Handles system operator commands and initiates other VSE/POWER tasks, or allows stopping of tasks.

WAIT TASK. Transfers the VSE/POWER partition to and from the wait state to meet system requirements.

RJE,BSC LINE MANAGER TASK. Controls line activities with remote terminals. The task is alive as long as VSE/POWER is active.

RJE,SNA MANAGER TASK. Controls the activation of transmission processing to and from a remote SNA workstation on a demand basis. The task is attached when the central operator issues a PSTART RJE,SNA command. The SNA manager also attaches a VSE/AF subtask in which the interface with VTAM is opened.

RJE,SNA LOGON TASK NO. 1. Initializes session work areas and does validity checking of logon request.

RJE,SNA LOGON TASK NO. 2. Establishes a session between VSE/POWER and a remote SNA workstation.

RJE,SNA LOGOFF TASK. Terminates a session between VSE/POWER and a remote SNA workstation.

RJE,SNA MESSAGE TASK. Sends messages to a remote SNA workstation.

SPOOL MANAGER TASK. Controls the activation and deactivation of the internal reader task and the spool/command manager list task. The task is attached during VSE/POWER initialization when SPOOL=YES is specified in POWER macro and detached at VSE/POWER termination.

READ TASK. (See Notes, item 2 on page 6.) Performs the first part of the read operation and transfer information from a peripheral reader to intermediate direct access storage. The operator may call for concurrent execution of as many read tasks as he has physical readers available. Each read task is therefore associated with a specific reader.

TAPE READ TASK. (See Notes, item 3 on page 6.) Performs the first part of a read operation and transfer information from a tape device to intermediate direct access storage.
The operator can call for the concurrent execution of as many tape read tasks as he has physical tape units available.

RJE,BSC READ TASK. (See Notes, item 2 on page 6.) Performs the read operation for a remote station. Each RJE,BSC read task has the standard name '1RDR' assigned to it. Different RJE,BSC read tasks are further distinguished by suffixing the line address to this standard name.

RJE,SNA READ TASK. Performs the read operation from a remote SNA workstation.

INTERNAL READER TASK. Performs the read operation for the PUTSPOOL VSE/POWER cross-partition communication macro interface.

EXECUTION READ TASK (See Notes, items 1 and 2 on page 6.) Performs the second part of the read operation and transfer information from intermediate direct access storage to meet the read requests of the user program. Each execution read task is associated with a specific partition. There can therefore exist as many execution read tasks as there are partitions controlled by VSE/POWER.

EXECUTION LIST TASK. (See Notes, items 1 and 3 on page 6.) Performs the first part of the list function and transfer information from the user program to either intermediate direct access storage or tape. There can exist as many execution list tasks as printers being spooled per partition, under VSE/POWER control. Each execution writer is associated with the partition for which it is spooling.

EXECUTION PUNCH TASK. (See Notes, items 1 and 3 on page 6.) Performs the first part of the punch function and transfer information from the user program to either intermediate direct access storage or tape. There may exist as many execution punch tasks as punches being spooled per partition, under VSE/POWER control. Each execution punch task is associated with the partition for which it is spooling.

LIST TASK. (See Notes, item 3 on page 6.) Performs the second part of the list operation and transfer information from intermediate direct access or tape storage to the printer. The operator may call for concurrent execution of as many list tasks as he has physical printers available. Each list task is therefore associated with a specific printer.

RJE/BSC LIST TASKS. (See Notes, item 3 on page 6.) Performs the list operation for a remote BSC work/workstation. Only one list task may be active per line. Each RJE,BSC list task has the standard name '1LST' assigned to it. Different RJE,BSC list tasks are further distinguished by suffixing the line address to this standard name.

RJE,SNA LIST TASK. Performs the list operation to a remote SNA workstation.

SPOOL/COMMAND MANAGER LIST TASK. Performs the list/punch retrieval (GETSPOOL) and the command invocation (CTLSPOOL) for the VSE/POWER spool manager interface.

PUNCH TASK. (See Notes, item 3 on page 6.) Performs the second part of the punch function and transfers information from intermediate direct access storage or tape to the punch. The operator may call for concurrent execution of as many punch tasks as he has physical punches available. Each punch task is therefore associated with a specific punch.

RJE,BSC PUNCH TASK. (See Notes, item 3 on page 6.) Performs the PUNCH operation for a remote BSC workstation. Each RJE,BSC punch task has the standard name '1PUN' assigned to it. Different RJE,BSC punch tasks are further distinguished by suffixing the line address.

RJE,SNA PUNCH TASK. Performs the punch operation to a remote SNA workstation.

ACCOUNT TASK. Supports the VSE/POWER job accounting option (together with VSE/AF JAI). It gives the user the option to either save the account file on another medium (tape, disk, cards) or delete the contents of the account file. The contents and format of the account records are not checked or changed by this task.

STATUS TASK. Scans the queue file, POFFLOAD or spool tapes, or network definition table and prints the status display or node information respectively on SYSLOG, a line printer, a workstation printer, builds a LST queue entry containing the desired information, or sends the queue status display / node information back to the originating node.
The Status task also performs the dumping of the storage copy of the queue file residing in VIO or partition GETVIS.

OFFLOADING TASK. Performs one of the following functions:

- Saves queue entries on tape, or
- Stores the contents (backup) of an entire class chain or queue onto tape, or
- Restores saved queue entries, selectively if desired, from POFFLOAD or spool tape to VSE/POWER queue.

The operator can call for the concurrent execution of as many off-loading tasks as he has physical tape units available.

TIMER TASK. Supports the time-sharing approach used by the Shared Spooling function. Interfaces with a VSE/AF subtask that handles the timer intervals.

NOTIFY TASK. Controls the transfer of messages destined for a particular subsystem, such as VSE/ICCF or VSE/DSNX. The task is attached by the SAS master task when the first 'notify' connection request is encountered; the task is detached at VSE/POWER termination.

PNET DRIVER TASK. Controls all activities on a PNET BSC/CTC or SDLC communication line. Processing is performed on a demand basis. The task is attached when the first PSTART PNET, nodeid is entered and detached when the last node is disconnected (signed off). The Network Driver also attaches a VSE/AF subtask in which the interface with VTAM is opened, when the first PSTART for a node is given using an SDLC communication line.

TRANSMITTER TASK. Transmits job or output to another node in the network. Up to eight transmitters can be active at a time for any node currently connected. There can be a mixture of job and output transmitters active concurrently. The task is active as long as there are jobs or output eligible for transmission.

RECEIVER TASK. Receives either job or output from another node in the network. Up to eight receivers, which may be a mixture of job or output receivers, can be active at a time for any node which is currently connected. The receiver task is attached only for the duration of the transmission of one job or output.

CONSOLE TRANSMITTER TASK. Sends messages and commands to another node in the network. The task exists as long there are messages or commands to send.

CONSOLE RECEIVER TASK. Receives messages and commands from another node in the network. The task is associated with a specific node and is attached when the node is started and exists as long as the connection to that node exists.

CONNECT TASK. Establishes a SNA session between VSE/POWER and another node in the network. The task is attached either when a PSTART nodeid is entered by the central operator, in which case the task acts as primary application, or when a BIND is received, in which case the task acts as secondary application.

DISCONNECT TASK. Terminates a SNA session between VSE/POWER and another node in the network acting as a primary or secondary application.

SPOOL ACCESS SUPPORT MASTER TASK. Acts as a watchdog and waits for connection requests from other partitions; it controls the activation of the notify task and SAS user tasks. The task is attached during VSE/POWER initialization and detached at VSE/POWER termination.

SPOOL ACCESS SUPPORT USER TASK. Performs each function request from a SAS user. The function can be to:

- Spool jobs to the VSE/POWER RDR or XMT queue.
- Spool list or punch data to the output queues.

- Retrieve job/output data from the various VSE/POWER queues.
- Manipulate queue entries in the various VSE/POWER queues.
- Issue VSE/POWER commands or to send messages.
- Retrieve fixed format Job Completion messages queue for a user

The task is attached by the SAS master task when a new XPCC connection is established and detached on demand by the SAS user or the system operator by means of the PSTOP command.

DEVICE SERVICE TASK. Services a Device Driving System (DDS) and transfers output data from the VSE/POWER spool files to the DDS processing the external output device. The task is attached by the command processor when a PSTART DEV command is given and terminated as result of a PSTOP DEV command.

TIME EVENT SCHEDULING TASK. Calculates the time interval for the first queue entry in the wait for run subqueue and waits for its expiration. After expiration it removes the queue entry from the wait for run subqueue and chains it to the really dispatchable class chain. The task is attached during VSE/POWER initialization and detached at VSE/POWER termination time.

HEARTBEAT TASK. Watches in an unattended node environment over VSE/OCCF. If VSE/OCCF terminates abnormally the heartbeat task forces VSE/POWER termination via PEND IMM and indicates REIPL to do. The task is attached by the spool access support master task if the environment is unattended and the application id is SYSOCCF. The task is detached at VSE/POWER termination time or if the connection is terminated normally or abnormally.

DYNAMIC PARTITION SCHEDULING TASK. Operates as a static 'hyper'- execution reader task. It allocates dynamic partitions for queue entries whose class is defined in the dynamic class table. The dynamic partition scheduling task is attached during VSE/POWER initialization. It is detached during VSE/POWER termination.

Notes:

1. Execution read tasks, execution list tasks, and execution punch tasks are collectively referred to as “execution processor” tasks.
2. Read, list and punch tasks are collectively referred to as “read/write” tasks.
3. Each read task is divided into two parts, physical reader (PR), tape reader (SY), BSC reader (BR) and/or 3540 reader (ER), which performs the device-dependent functions related to data collection from a specified device or family of devices, and logical reader (LR), which performs the logical functions related to entering input data into the VSE/POWER data file and inserting a new queue entry into the correct position in the VSE/POWER queue file. These two parts are linked by means of a high-level logical record interface.

   Similarly, each list task is divided into Physical List (PL), BSC Writer (BW) or Physical Punch (PP), and Logical Writer (LW). The physical part of a task performs device-dependent functions for printer or punch, respectively. The Logical Writer retrieves data from the list queue or the punch queue, as required. In each case a high-level logical record interface is defined to connect the two parts of the task.

Figure 1 shows the relationship of the user program to the VSE/POWER partition and tasks and to the VSE/AF supervisor.

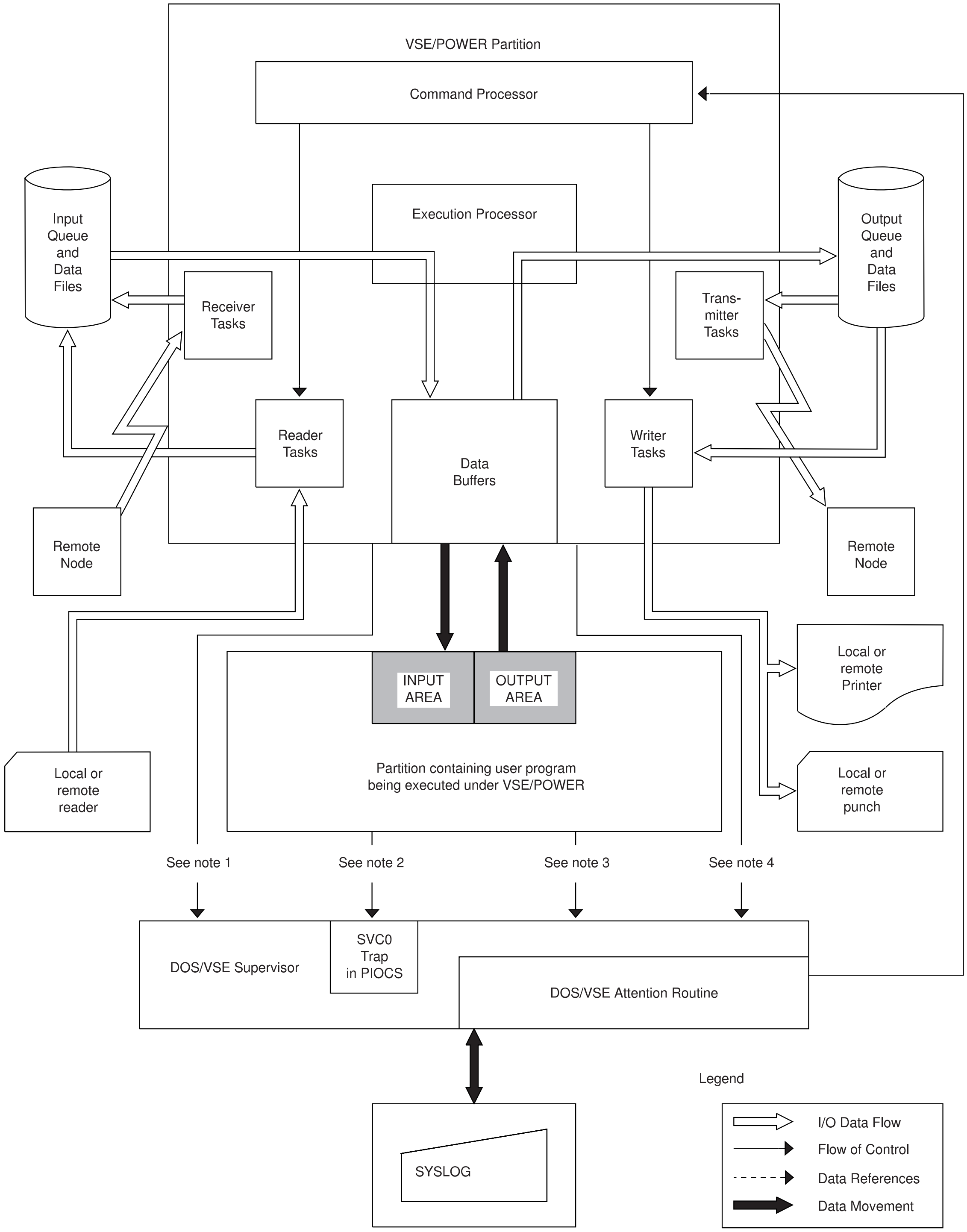


Figure 1. Relationship Between VSE/POWER, VSE/AF, and the Program Running under the Control of VSE/POWER

Notes:

1. SVC 0 for I/O to VSE/POWER files.
2. SVC 0 from user partition unit record devices converted to I/O to a spooling device on VSE/POWER partition.
3. SVC 7 issued by VSE/POWER.
4. SVC 0 for I/O from VSE/POWER files

VSE/POWER Direct Access Files

VSE/POWER files can be either on Count-Key-Data (CKD) or Fixed Block Architecture (FBA) devices. A mixture is also possible, however the extents of the Data File must all reside on the same disk type.

For a list of disk types available to contain the VSE/POWER 7.1 spool files refer to *z/VSE 3.1 Planning*.

For more information on the organization of the VSE/POWER files, refer to the *VSE/POWER Administration and Operation*.

Communication with VSE/POWER

VSE/POWER Operator Commands

VSE/POWER provides operator commands that allow the central-system operator and the remote-terminal operator to communicate with the system. Following types of commands are provided:

- Task-management commands, which allow the operator to initiate and terminate VSE/POWER tasks (except spool management tasks). For RJE, task management commands are only applicable to the RJE writer task. The RJE reader task is started whenever the workstation is ready to send. Its operation is controlled by the system.
- Queue-management commands, which allow the operator to display and modify the contents of VSE/POWER queue entries. Queue-management commands are only applicable to users that have the correct authority.
- List-control commands, which allow the operator to perform certain device-dependent operations on line printers.
- Workstation-control commands, which allow the remote operator to initiate and terminate VSE/POWER RJE tasks.
- Network-control commands, which allow the operator to perform certain operations to control the functioning of the network system.
- External device control commands, which allow the operator to initiate, control and terminate external devices.

For further information on VSE/POWER operator commands refer to *VSE/POWER Administration and Operation*.

Job Entry Control Language

VSE/POWER provides a job entry control language (JECL) to assist the user in delimiting jobs to the system and to allow him to specify special requirements that may apply to particular jobs. JECL supplements but does not replace the job control language (JCL) provided by VSE/AF itself. The JCL statements required for normal VSE/AF system operation are also required when operating under VSE/POWER.

For a detailed description of the VSE/POWER JECL statements, refer to *VSE/POWER Administration and Operation*.

Format of VSE/POWER Operator Messages

Messages sent by VSE/POWER to SYSLOG, SYSLST, or to a terminal may have the following formats:

```
1QnnI or 1RnnI or 1VnnI  (information-type message)
1QnnD                    (decision-type message)
1QnnA or 1RnnA or 1VnnA  (action-type message)
```

where:

Q VSE/POWER general-message indicator.

R VSE/POWER message indicator for messages issued by RJE,BSC, the command-processor tasks, the Shared Spooling function and networking.

V VSE/POWER message indicator for messages issued by RJE,SNA tasks.

nn   message-identification number. (May also include alpha characters.)

I-type messages are for the operator's information only; no response is required. Processing continues normally.

D-type messages require an immediate reply from the operator.

A-type messages require some action from the operator, such as mounting a tape. A-type messages for remote workstations are directly displayed on the remote printer. The VSE/POWER task issuing the message is put in the wait state.

Messages issued by VSE/POWER are listed in “Message Reference” on page 401 and are further described in the manual *VSE/ESA Messages and Codes*.

Environmental Requirements

Programming Requirements

The programming requirements for VSE/POWER are as follows:

1. For Basic VSE/POWER

   - The following phases must be cataloged in a library.

     IPW$$AQ
     IPW$$AS
     IPW$$AT
     IPW$$CA
     IPW$$CB
     IPW$$CC
     IPW$$CD
     IPW$$CE
     IPW$$CF
     IPW$$CG
     IPW$$CH
     IPW$$CI
     IPW$$CJ
     IPW$$CL
     IPW$$CLD
     IPW$$CM
     IPW$$CO
     IPW$$CP
     IPW$$CR
     IPW$$CS
     IPW$$CT
     IPW$$CU
     IPW$$CV
     IPW$$CX
     IPW$$CY
     IPW$$DD
     IPW$$DP
     IPW$$DQ
     IPW$$DS
     IPW$$DT
     IPW$$ER
     IPW$$FQ
     IPW$$GD
     IPW$$IC
     IPW$$ID
     IPW$$IP
     IPW$$I1
     IPW$$I2
     IPW$$I3
     IPW$$I4
     IPW$$I5
     IPW$$I7
     IPW$$LO
     IPW$$LR
     IPW$$LU
     IPW$$LW
     IPW$$MM
     IPW$$MS
     IPW$$MX
     IPW$$NQ
     IPW$$NS
     IPW$$NU
     IPW$$OE
     IPW$$OF
     IPW$$OP
     IPW$$OT
     IPW$$PC
     IPW$$PD
     IPW$$PL
     IPW$$PP
     IPW$$PR
     IPW$$PS
     IPW$$PS1
     IPW$$Q1
     IPW$$RQ
     IPW$$RY
     IPW$$SC
     IPW$$SQ
     IPW$$SY
     IPW$$TQ
     IPW$$TR
     IPW$$TV
     IPW$$T1
     IPW$$XH
     IPW$$XJ
     IPW$$XM
     IPW$$XRE
     IPW$$XT
     IPW$$XTC
     IPW$$XTG
     IPW$$XTP
     IPW$$XTM
     IPW$$XTS
     IPW$$XWE

   - The following macros must be cataloged in a library.

     POWER
     PWRSPL
     IPWSEGM
     IPW$MXD
     SEGMENT

   - The number of entries in the LUB table (in the VSE/AF supervisor), belonging to the VSE/POWER partition must be large enough to accommodate all reader and writer tasks that may be running concurrently. Programmer LUB's SYS000 through SYS034 are reserved for the account, queue, and data files, and queue file re-allocation.

   - I/O Files

     The queue file and data file must be assigned to a spooling device as required:

     Queue file - SYS001
     Data file - SYS002 through SYS033

     A queue file extent, with the name IJQFILE, and data file extent(s), with the name IJDFILE, must be defined by DLBL/EXTENT statements.

2. For VSE/POWER RJE/BSC (optional)

   - The following phases must be cataloged in a library.

     IPW$$BM
     IPW$$BR
     IPW$$BW
     IPW$$LM

   - The following macros must a cataloged in a library.

     PLINE
     PRMT

3. For VSE/POWER RJE/SNA (optional)

   - The following phases must be cataloged in a library.

     IPW$$IB
     IPW$$LF
     IPW$$LH
     IPW$$LN
     IPW$$MP
     IPW$$OB
     IPW$$OC
     IPW$$SN
     IPW$$VE

   - The following macros must be cataloged in a library.

     PRMT
     PCPTAB

4. For VSE/POWER Networking (optional)

   - The following phases must be cataloged in a library.

     IPW$$BS
     IPW$$CAC
     IPW$$CN
     IPW$$CPF
     IPW$$CPS
     IPW$$IN
     IPW$$LD
     IPW$$LD1
     IPW$$LD2
     IPW$$LD3
     IPW$$LD4
     IPW$$LD5
     IPW$$NC
     IPW$$NK
     IPW$$NM
     IPW$$NP
     IPW$$NR
     IPW$$NR2
     IPW$$NT
     IPW$$SD
     IPW$$SE
     IPW$$SR
     IPW$$SS
     IPW$$S1
     IPW$$S2
     IPW$$S3
     IPW$$TD
     IPW$$TS

   - The following macros must be cataloged in a library.

     PNODE
     PLINE (BSC only)

5. For VSE/POWER Accounting (optional)

   - If the account file is on a CKD device, the following phases must be cataloged in a library.

     IPW$$BA
     IPW$$GA
     IPW$$PA
     IPW$$SA

   - If the account file is on an FBA device, the following phases must be cataloged in a library.

     IPW$$BA
     IPW$$GF
     IPW$$PF
     IPW$$SF

   - The following macros must be cataloged in a library.

     PACCNT
     PUTACCT

   - The required job accounting specifications must be given at VSE/AF IPL time.
   - An account file must be assigned to SYS000 on a disk device. Account file space must be defined for IJAFILE by the DLBL/EXTENT statements.

6. For VSE/POWER Spool manager (CTLSPOOL/GETSPOOL/PUTSPOOL) support (optional)

   - The following phase must be cataloged in a library.

     IPW$$SM

   - The following macros must be cataloged in a library.

     CTLSPOOL
     GETSPOOL
     PUTSPOOL
     SPL

7. For VSE/POWER SLI Support (optional)

   - The following phase must be cataloged in a library.

     IPW$$SL

8. For VSE/POWER Shared Spooling function (optional)

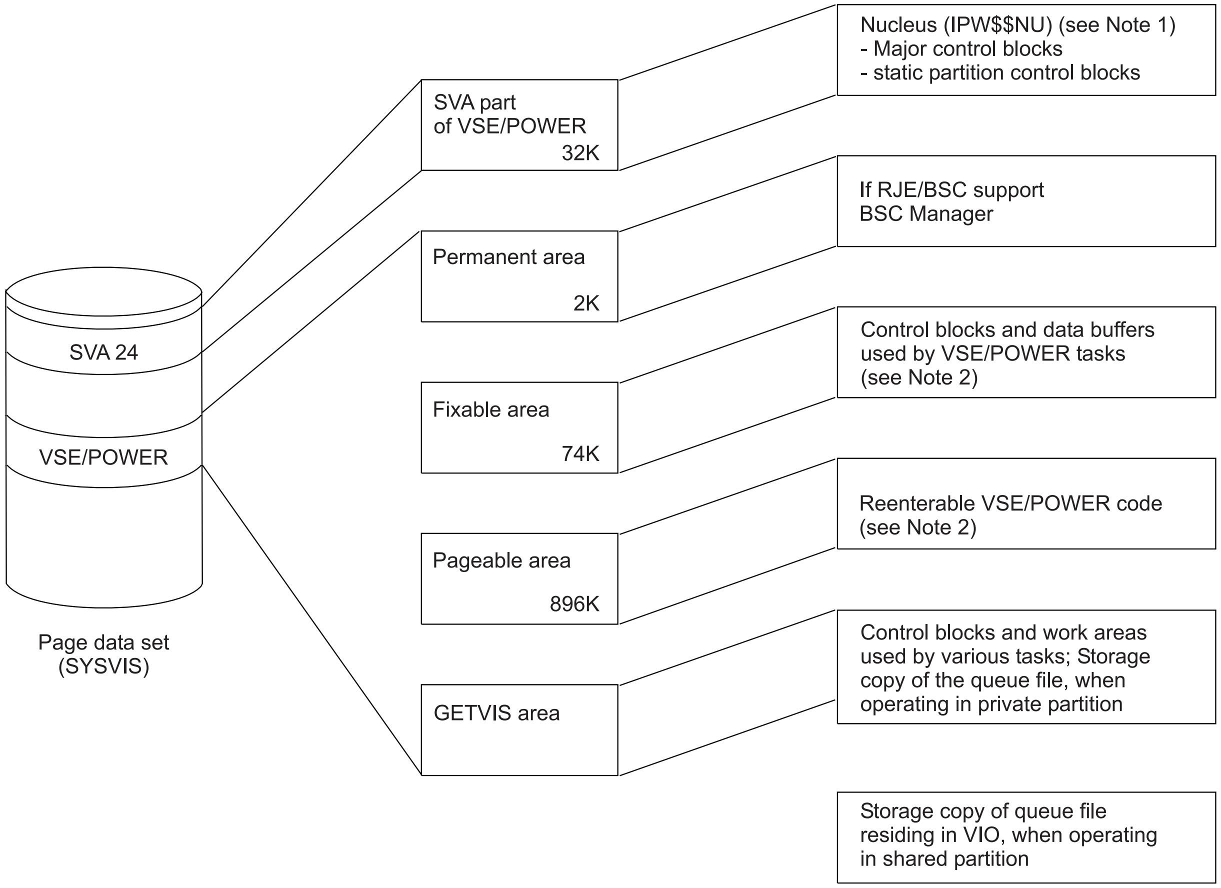
   - The following phases must be cataloged in a library.

     IPW$$TI
     IPW$$CRE

Storage Requirements and Allocations

Virtual Storage

The virtual-address space of the VSE/POWER partition consists of five major areas as shown in Figure 2.


Nucleus (IPW$$NU) (see Note 1)
- Major control blocks
- static partition control blocks
SVA part
of VSE/POWER
32K
If RJE/BSC support
BSC Manager
Permanent area
2K
SVA 24
Control blocks and data buffers
used by VSE/POWER tasks
(see Note 2)
Fixable area
74K
VSE/POWER
Reenterable VSE/POWER code
(see Note 2)
Pageable area
896K
Page data set
(SYSVIS)
Control blocks and work areas
used by various tasks; Storage
copy of the queue file, when
operating in private partition
GETVIS area
Storage copy of queue file
residing in VIO, when operating
in shared partition

Minimum SIZE of virtual partition (no optional support) = 896K
plus size of real partition + min. partition getvis area (48K)
Note 1 - See PDISPLAY STATUS "SVA USED BY VSE/POWER"
for Nucleus with 11 Static Partitions
Note 2 - See Administration & Operations Manual "Processing Requirements"

Figure 2. Basic Organization of the VSE/POWER Partition

- The SVA part of VSE/POWER is 32K and is permanently fixed in the SVA. This is done during initialization of VSE/POWER and freed when VSE/POWER is terminated. The SVA part contains the nucleus (IPW$$NU) with the control address table ("CAT", also called the permanent area) and the POWER partition control blocks for static partitions.

  The POWER partition control blocks for dynamic partitions are also allocated in the SVA but not in these 32K.

- If RJE,BSC support is used, the permanent area is 2K and permanently fixed in the real partition during initialization of VSE/POWER. These pages are freed only when VSE/POWER is terminated. If RJE,BSC support is not used, these 2K belong to the fixable area.

- The fixable area consists of pages that are fixed when a task is started and freed when they are no longer required for the completion of this task.

- The pageable area consists of pages that are allowed to be paged out whenever VSE/AF requires additional real storage.
- The GETVIS area contains logical data areas, queue record areas, control blocks and work areas used by RJE SNA, PNET, VSE/AF service routines, SYSIN tape support, printer setup processing (3800 or 3200), fixed format Job Completion messages queues, notify message queue and I/O buffers for accounting if the account file is on an FBA device.

  If VSE/POWER has been activated in a private partition, the storage copy of the queue file is placed into the partition GETVIS area also.
- VIO or partition GETVIS area contains the storage copy of the queue file while VSE/POWER is active. The size of the area depends on the size of the queue file. 384 bytes are allocated for each queue record. The allocation for the VIO area is rounded up to the next multiple of 64K.

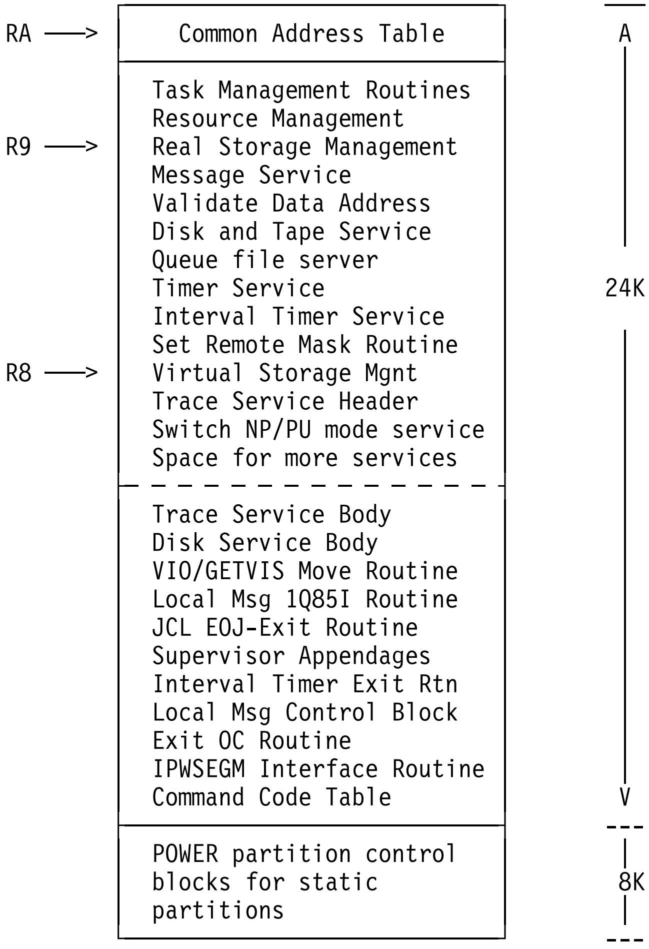


Figure 3. SVA Part of VSE/POWER

Real Storage

The minimum real-address space must be equal to the size of the permanent area (2K, if RJE BSC is generated) plus the fixable area. The size of the fixable area (minimum = 74K) depends on the number of tasks active at any time and the control blocks and work areas used by these tasks.

For a description of how to calculate VSE/POWER storage requirements, refer to *VSE/POWER Administration and Operation*.

Hardware Supported

Machine Requirements

Any processor supported by VSE.

Devices Supported

The devices for which support has been introduced in VSE/POWER are listed in Figure 4 and may exist until deleted. For a complete list of officially supported devices refer to *VSE/ESA System Control Statements*.

Figure 4. Devices Supported by VSE/AF

Readers	Printers[4]	Punches	Spooling Devices	Terminals
1442	1403	1442	2400 series[1] [2]	2770[3]
2501	3200	2520	3400 series[2]	2780
2520	3203/4/5	2540	3350	3741
2540	3211	2596	3380	3771[5]
2596	3262/1/2	3525	3390	3773[5]
3505	3289/4		3480	3774[5]
3525	3800		3490	3775[5]
3540	4245		8809	3776[5]
	4248		9300 series	3777-1
	6262		FBA	3780[3]
				3790[6]

Notes:

1. The IBM 2495 tape cartridge reader does not belong to this series.
2. List, punch, tape reader and off-load devices. For 7-track tape units the data conversion feature is required.
3. The following I/O devices are supported by RJE,BSC.

   IBM 545 Punch (Model 3 or 4)
   IBM 2213 Printer (Model 1 or 2)
   IBM 2502 Card Reader (Model A1 or A2) on the 2770
   IBM 2203 Printer (Model A1 or A2)
   IBM 3781 Punch on the 3780

   Teleprocessing control units supported by RJE,BSC are:

   IBM 2701 Data Adapter Unit with SDA (Type 2)
   IBM 2703 Transmission Control Unit
   IBM 3704 Communications Adapter in 2703 emulation mode
   IBM 3705 Communications Adapter in 2703 emulation mode
   IBM 372x Communications Adapter in 2703 emulation mode
   Integrated Communications Adapter in the various models

   Restrictions:

   - TP connections must be point-to-point on switched or non-switched lines.
   - Multipoint connections are not supported.

- Terminals and control units having the multipoint line control or multipoint data link control features are prohibited. (Connecting such a terminal or control unit to the POWER/RJE,BSC system will cause continuous error recovery processing.)

4. The Universal Character Set Buffer (UCB) and Forms Control Buffer (FCB) features are supported by VSE/POWER. The execution processor will accept UCB and FCB load requests from the various supported partitions for appropriate action at list time. On encountering an FCB load command, the execution processor will update the internal buffer representation to reflect the new buffer.

5. The following I/O devices are supported by RJE,SNA:

   IBM Magnetic Diskette Storage (3774, 3775, 3776)
   IBM 2502 Card Reader (3774, 3775, 3776)
   IBM 3501 Card Reader (3771, 3774, 3775, 3776)
   IBM 3521 Card Punch (3771, 3774, 3775, 3776)
   IBM 3784 Printer (3774)

   VSE/POWER does not control SDLC lines. The SNA terminals may be connected to VTAM and the NCP on any communication media supported by VTAM.

6. The 3790 support is limited to that comprising the 3790 RJE facility.

Chapter 2. Method of Operation

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 5) 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.

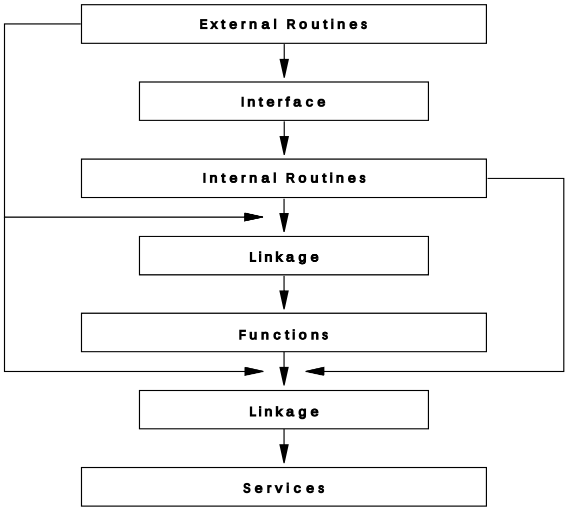


Figure 5. Hierarchic Structure of VSE/POWER

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 - Nucleus Base Register

Register 10 points to the beginning of the nucleus which contains first the control address table (CAT), the main VSE/POWER control block. R10 secures also addressability for task management and task services contained in the first part of the VSE/POWER nucleus. 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/AF 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 or in case of an SAS user task in the XP work area. The external save area contains in word 1 the address of the next (internal) save area and word 2 contains the address of the previous save area, if there is any. The internal save area contains in word 1 the address of the calling task TCB and the second word contains the address of the previous (external) save area. Figure 6 illustrates the relationship.

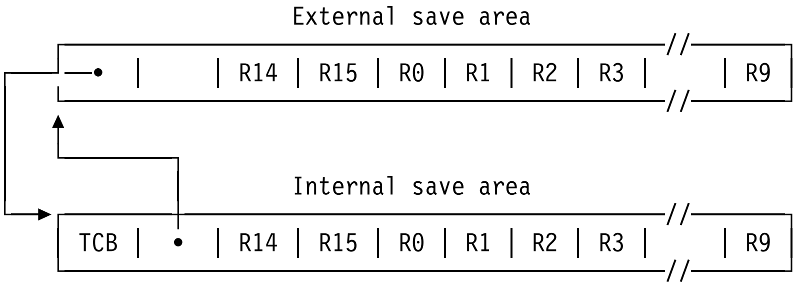


Figure 6. Relationship of Internal and External Save Area

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 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 table (CAT) for the base VSE/POWER routines or in the PNET master control block for all PNET routines.

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 7 shows the various usages of the registers 0 through 3.

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

Macro	R0		R1		R2		R3		Other
	Before	After	Before	After	Before	After	Before	After	
IPW$ATT	return		TCB	TCB					
IPW$DET	ECB		TCB						
IPW$WFE			ECB						
IPW$WFI									
IPW$WFO									
IPW$WFL							RCB		
IPW$WFM			list						
IPW$WFQ			list						R12=return address
IPW$WFC			CCB or						
IPW$WFS			ECB						
IPW$WFD									
IPW$WFX			List						
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$RDQ	return		IORW						
IPW$RDD	return		IORW						
IPW$WTQ	return		IORW						
IPW$WTD	return		IORW						
IPW$WTT	return		TRW						
IPW$RDT	return		TRW						
IPW$CTT	return		TRW						
IPW$RDC				TOD	return				
IPW$VDA		return code			return				R6=PDB R8=CCWaddr.
IPW$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 of area or zero	return				R4=Pointer to anchor

Figure 7 (Part 1 of 2). Contents of Registers when a Service is Invoked

Macro	R0		R1		R2		R3		Other
	Before	After	Before	After	Before	After	Before	After	
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$NTY	pointer to target node name		address NMR		return				
IPW$GQR			IORW		return				
IPW$MQR			IORW		return				
IPW$WQR			IORW		return				
IPW$TDM		unpred.	NP/PU option		return				

Figure 7 (Part 2 of 2). Contents of Registers when a Service is Invoked

Chapter 3. Program Organization

This section describes the program organization of VSE/POWER. It outlines the logical structure of the VSE/POWER Program Product, presenting overviews of all internal operations and indicating the relationships between the various tasks and routines.

The following topics are discussed:

Code organization explains the VSE/POWER code and storage structure and lists the internal macros.

Initialization and termination gives an overview of the phases that handle startup and shutdown of VSE/POWER processing.

VSE/POWER multitasking explains the principles of task selection, of starting a task, and of terminating a task.

Reader, execution processor, and writer tasks shows the data flow through the spooling process, and highlights the work done by the various phases related to these tasks.

Dynamic Partition Support describes enabling, modifying and interplay of dynamic partitions.

The Spooling Process describes spooling to and from the queue file and the data file.

Running in ESA mode describes spooling to and from the queue/data file when VSE/POWER is running on MODE=ESA supervisor.

Multiprocessor Support describes how this support can be activated and exploited in VSE/POWER, and how it is implemented internally.

Services describes the routines of the nucleus phase.

Miscellaneous tasks and functions describes various tasks and functions that are not readily associated with the above areas.

Command processor gives an overview of command processing; how the command processor is invoked, and what actions are taken.

VSE/POWER job accounting describes the Account functions and the save account task.

VSE/POWER Networking gives an overview of how the Networking function works.

Remote job entry highlights the essentials of RJE,BSC and RJE,SNA.

Appendages lists the routines in the nucleus phase that are extensions of the VSE/AF system control programs.

VSE/POWER Shared Spooling gives an overview of how the Shared Spooling function works.

Spool Access Support Interface describes the various functions of the interface and gives an overview of how the SAS interface works.

External Device Support gives an overview of how that support works.

Code Organization

Storage Structure

The address space of the virtual VSE/POWER partition is composed of four major areas, each containing an integral number of pages:

- The permanent area (CAT)
- The fixable area
- The pageable area
- The GETVIS area

Additional to the areas in the partition, VSE/POWER needs an area fixed in the SVA.

The SVA part of VSE/POWER is fixed during VSE/POWER initialization and contains the VSE/POWER nucleus with major control tables and routines such as control address table (CAT). It also contains the partition control blocks of the static partitions. The pages are freed when VSE/POWER is terminated.

The partition control blocks of the dynamic partitions are also located in the SVA. They are allocated by the supervisor during partition allocation and freed during partition deallocation.

The first page of the VSE/POWER virtual address area is fixed in real storage as soon as the VSE/POWER system begins execution, and remains fixed till the system is terminated. It contains the RJE,BSC I/O Monitor if the RJE function is generated. If the RJE function is not generated the page is added to the fixable area.

The fixable area is contained in the second group of pages within the VSE/POWER virtual address area. These provide the necessary address space for dynamically-structured control areas and for physical data buffers used by the VSE/POWER tasks. The size of this area depends on the amount of real address space that the user has assigned to the VSE/POWER partition. The pages within the area are dynamically fixed when reserved for specific task use and freed when no longer required. At any point in time certain pages within the area will be fixed while others are free; the necessary page fixing and freeing is controlled by the real storage management service of VSE/POWER. Some of the control blocks such as the disk management block (DMB) and module control blocks (MCB) are allocated at VSE/POWER start up time and exist as long as VSE/POWER is active. See Figure 142 on page 440.

The pageable area is contained in the third group of pages within the VSE/POWER virtual address area. These contain the remaining phases of the VSE/POWER code and may be paged at any time the system requires additional real storage. The size of this area depends on the particular VSE/POWER phases required for system execution; this in turn depends upon the execution options selected by the user (accounting, RJE, reader exit, PUTSPOOL/GETSPOOL/CTLSPOOL support, shared spooling, SLI facility, PNET and spool access support).

The GETVIS area is contained in the remaining pages within the VSE/POWER virtual address area. These contain control blocks and work areas used by the VSE/POWER tasks and the storage copy of the queue file when running in a private partition. The pages may be paged out at any time the system requires additional real storage. The size of the GETVIS area depends on the size of the logical data buffers, which is determined by the value specified in the DBLK parameter of the POWER macro, the maximum number of tasks active at any time with their storage requirements, and the number of queue records as determined by the // EXTENT statement of the IJQFILE.

The GETVIS-24 area is used for the following purposes:

- RJE,SNA operation
- 3200/3800 printer setup processing

- Logical data areas associated with each task
- Queue record areas associated with each task
- Input buffer for SYSIN tape support
- PNET SNA transmission buffers
- Message queue(s)
- Input/Output buffer for Accounting (FBA only)
- PNET operation
- SAS task work areas
- Exit phases (via PLOAD)
- Exit work areas
- VSE/POWER phases (via PLOAD)
- Queue file storage copy when running in a private partition, and no partition Getvis-31 exists.

The GETVIS-31 area (provided by sufficient ALLOC) is used for:

- Queue file storage copy when running in a private partition. If the Getvis-31 area cannot house the storage copy totally, then the copy stretches over the 16 MB line.

The areas are allocated by the appropriate tasks when needed and freed when the tasks terminate or when no longer needed.
Some of the control blocks, such as the network definition table (NDT) and the queue file storage copy are allocated at VSE/POWER initialization time and exist as long as VSE/POWER is active.

The GETVIS area is logically divided into the following pools:

- General pool
- Message/command pool
- PNET pool
- RJE,SNA pool (general)
- RJE,SNA WACB and compaction table pool

The page frames within real processor storage that are occupied by VSE/POWER at any time are divided into two groups:

- The first group of page frames is obtained from within the VSE/AF page pool and contain pages of code from the VSE/POWER pageable area which are currently being referenced for instruction execution. Page frames within this group remain part of the page pool and are susceptible to system paging.
- The second group of page frames is withdrawn from the VSE/AF page pool and contain, firstly, the pages of the VSE/POWER permanent area and, secondly, those pages of the VSE/POWER fixable area which have been fixed in real storage by the VSE/POWER real storage management service.

Note: The pages of the VSE/POWER fixable area which have not been fixed in real storage by VSE/POWER do *not* occupy real storage in any sense.

Code Structure

The code of VSE/POWER consists of External Routines, Internal Routines, Functions, Services, and Appendages.

External Routines: External routines provide task support at the highest level of the system. Each external routine consists of a single phase which is physically located in the VSE/POWER pageable area.

The following external routines are provided:

IPW$$BR	RJE,BSC Reader
IPW$$BW	RJE,BSC Writer
IPW$$CM	Command Processor Root Phase
IPW$$ER	3540 Diskette Reader
IPW$$IB	RJE,SNA Inbound Processor
IPW$$LD	Network PNET Driver
IPW$$LF	RJE,SNA Logoff Processor
IPW$$LH	RJE,SNA Logon Processor No. 1
IPW$$LM	RJE,BSC Line Manager
IPW$$LN	RJE,SNA Logon Processor No. 2
IPW$$MP	RJE,SNA Message Processor
IPW$$NS	Notify Support
IPW$$OB	RJE,SNA Outbound Processor
IPW$$OC	Outbound Compaction Manager
IPW$$OF	Offload Queues Routine
IPW$$PL	Physical List
IPW$$PP	Physical Punch
IPW$$PR	Physical Reader
IPW$$PS	Perform queue/node display
IPW$$NR	Network Receiver
IPW$$NR2	Network Receiver Part 2
IPW$$NT	Network Transmitter
IPW$$SA	Save Account
IPW$$SD	PNET SSL SD Subtask
IPW$$SE	PNET,SNA VTAM Exit Routines
IPW$$SF	Save Account for Account File on FBA device
IPW$$SM	Internal Reader Spool Command Manager
IPW$$SN	RJE,SNA Manager
IPW$$SY	SYSIN Tape Reader Routine
IPW$$S1	PNET,SNA OPEN/CLOSE Subtask
IPW$$S2	PNET,SNA Connect Processor
IPW$$S3	PNET,SNA Disconnect Processor
IPW$$TD	PNET TCP/IP TD Subtask
IPW$$TI	Timer Task
IPW$$TV	Time Event Scheduling Task
IPW$$VE	RJE,SNA VTAM Exit Routines
IPW$$XH	Heartbeat Task
IPW$$XM	Spool Access Support Master Task
IPW$$XT	Spool Access Support Main Routine
IPW$$XTC	Spool Access Support CTL-Function Routine
IPW$$XTG	Spool Access Support GET-Function Routine
IPW$$XTP	Spool Access Support PUT-Function Routine
IPW$$XTM	Spool Access Support GCM-Function Routine
IPW$$XTS	Spool Access Support Subroutines

External Macros: External macros provide system generation and API support. The following external macros are provided:

Generation Macros

POWER	System Generation Macro
PACCNT	Accounting DSECT Generation Macro
PCPTAB	RJE,SNA Compaction Table Generation Macro
PRMT	RJE Hardware Generation Macro
PLINE	RJE,BSC Line Hardware Generation Macro
PNODE	PNET Network Definition Table Generation Macro

API Interface Macros

PUTACCT	Execution Accounting Record Append Macro
PWRSPL	SAS PWRSPL Update and DSECT Generation Macro
CTLSPOOL	Spool Manager Support - Control Macro
GETSPOOL	Spool Manager Support - Get Macro
PUTSPOOL	Spool Manager Support - Put Macro
SPL	Spool Manager Support - DSECT Macro
SEGMENT	Segmentation Macro (single threaded)
IPWSEGM	Segmentation Macro (enhanced multiple threaded)

Internal Routines: Internal routines provide task support at a level below external routines, which communicate with them by means of the Interface Macro Instructions described below. Each internal routine consists of a single phase which is physically located in the VSE/POWER pageable area.

The following internal routines are provided:

IPW$$DP	Dynamic partition scheduler
IPW$$LO	Logical Output Routine
IPW$$LR	Logical Reader
IPW$$LW	Logical Writer
IPW$$XRE	Execution Reader
IPW$$XWE	Execution Writer

Functions: Functions provide support for operations common to two or more routines; they are to be regarded as high-level subroutines capable of concurrent execution, and are invoked by means of the Function Macro Instructions described below. Each function consists of a single phase which is physically located in the VSE/POWER pageable area.

The following functions are provided:

IPW$$AQ	Add Queue Entry to Chain
IPW$$AS	Asynchronous Service
IPW$$AT	Abnormal Termination
IPW$$BA	Build Account Record Routine
IPW$$BM	RJE/BSC Monitor (placed in fixed storage)
IPW$$BS	Network Buffer Management Routines
IPW$$CA	PALTER Command Processor
IPW$$CAC	PACT Command Processor
IPW$$CB	PBRDCST Command Processor
IPW$$CC	PCANCEL Command Processor
IPW$$CD	PDISPLAY Command Processor
IPW$$CE	PEND Command Processor
IPW$$CF	PFLUSH Command Processor
IPW$$CG	PGO Command Processor
IPW$$CH	PHOLD Command Processor
IPW$$CI	PINQUIRE Command Processor
IPW$$CJ	PACCOUNT Command Processor
IPW$$CL	PDELETE Command Processor
IPW$$CLD	PLOAD Command Processor
IPW$$CN	PDRAIN Command Processor
IPW$$CO	POFFLOAD Command Processor
IPW$$CP	PSTOP Command Processor
IPW$$CPF	PFLUSH PNET Command Processor
IPW$$CPS	PSTART PNET Command Processor
IPW$$CR	PRELEASE Command Processor
IPW$$CRE	PRESET Command Processor
IPW$$CS	PSTART Command Processor
IPW$$CSG	PSEGMENT Command Processor
IPW$$CT	PRESTART Command Processor
IPW$$CU	PSETUP Command Processor
IPW$$CV	PVARY Command Processor
IPW$$CX	PXMIT Command Processor
IPW$$CY	PCOPY Command Processor
IPW$$DQ	Delete Queue Entry from Chain
IPW$$DS	Data Management Service Routines
IPW$$DT	Define Default Control Records and Tables
IPW$$FQ	Free Queue Entry
IPW$$GA	Get Account Record
IPW$$GD	Get Data Record
IPW$$GF	Get Account File on FBA Device
IPW$$IC	Invoke Command Processor
IPW$$ID	Process IDUMP 'In Flight' Request
IPW$$LU	Update LUB and PUB Tables
IPW$$MM	Message Module
IPW$$MS	Message Handler
IPW$$MX	Message Distributor / Modification

IPW$$NC	Network Composer
IPW$$NK	Network Compression/ Decompression
IPW$$NM	Network I/O Manager
IPW$$NP	Network Presentation Services
IPW$$NQ	Get Next Queue Entry from Chain
IPW$$OE	3540 Diskette Open
IPW$$OP	User Defined Output Parameter Processing Routine
IPW$$OT	Open/Close Tape
IPW$$PA	Put Account Record
IPW$$PC	Parameter Checker
IPW$$PF	Put Account Record for Account File on FBA Device
IPW$$PD	Put Data Record
IPW$$PS1	Print Queue Display Service Routine
IPW$$Q1	Allocate/Deallocate DBLK-Group Routine
IPW$$RQ	Reserve Queue Record
IPW$$RY	Queue-File Recovery
IPW$$SC	Scan Reader JECL Statement
IPW$$SL	Get Source-Statement-Library Record
IPW$$SQ	Queue Management Service Routines
IPW$$SR	PNET,SNA Send/Receive Manager
IPW$$TQ	Add Queue Entry to Wait for Run Subqueue
IPW$$TQI	Check Expiration of Due Date
IPW$$TR	Terminate VSE/POWER Task
IPW$$XJ	Scan Execution JECL Statement

Services: Services provide support for operations common to many routines and functions; they are to be regarded as low-level subroutines capable of concurrent execution, and are invoked by means of the Service Macro Instructions described below. Each service is coded as a separate segment; all of these segments are however physically located within the nucleus phase (IPW$$NU).

The following services are provided:

- Disk and Tape Service
- Queue File Service
- Message Service
  Local Message Service
  Remote Message Service
  Notify Service
  Nodal Message Service
- Remote Service
- Resource Management
- Storage Management
- Task Management
- Timer Service
- Interval Timer Service
- Validation Service
- Get Trace Entry Routine
- Virtual Storage Management
- Switch NP/PU Mode Service

Appendages: Appendages provide code which, though physically present in the nucleus phase (IPW$$NU), is logically part of the VSE/AF supervisor or of some other VSE component. Appendages may reference and update VSE/POWER tables and data areas but may not invoke any VSE/POWER routine, function, or service, and may not be invoked by them.

The following appendages are provided:

- Page Fault Appendages
- Attention Interface Appendage
- RJE/PNET Channel End Appendage
- Hot Reader Appendage
- SVC 0 / 3 Appendage
- SVC 90 and SVC 91 Appendage
- Timer Interval Exit Routine
- JCL End-of-Job Appendage

Internal Macro Instructions

Communication between external routines, internal routines, functions, and services is performed by means of VSE/POWER internal macro instructions. Macro Instructions are also provided to define the format of common tables and data areas, and to perform other miscellaneous functions.

There are five types of VSE/POWER internal macro instructions:

- Interface macros - see Figure 8.
- Function macros - see Figure 9 on page 33.
- Service macros - see Figure 10 on page 34.
- Definition macros - see Figure 11 on page 36.
- Miscellaneous macros - see Figure 12 on page 37.

Figure 8. Interface Macros

Macro	Purpose
IPW$OLI	Open logical interface
IPW$CLI	Close logical interface
IPW$PLR	Put logical record
IPW$GLR	Get logical record

Figure 9. Function Macros

Macro	**Purpose**
Queue management	
IPW$AQS	Add queue entry to chain
IPW$DQS	Delete queue entry from chain
IPW$FQS	Free queue entry
IPW$GQS	Get next queue entry from chain
IPW$IQS	Invoke queue management services
IPW$RQS	Reserve queue record
IPW$ITQ	Add/Delete queue entry from Wait for Run subchain or INIT Wait for Run subchain
Data management	
IPW$GDR	Get data record
IPW$IDS	Invoke data management services
IPW$PDR	Put data record
Account management	
IPW$CAF	Close account file (delete contents of account file if on FBA device)
IPW$OAF	Open account file (not required for FBA)
IPW$GAR	Get account record (not required for FBA)
IPW$PAR	Put account record
Other functions	
IPW$BUF	Invoke PNET Buffer management
IPW$CNC	Cancel VSE/POWER or terminate VSE/POWER task
IPW$GMS	Invoke general message service
IPW$GSL	Get source statement library record
IPW$IAS	Invoke asynchronous service
IPW$ICP	Invoke command processor
IPW$IDM	Invoke IDUMP 'In Flight' routine
IPW$IIS	Invoke queue display service routine
IPW$IOC	Invoke outbound compaction manager (IPW$$OC)
IPW$IOM	Invoke RJE,BSC I/O Monitor, PNET I/O Manager
IPW$IPS	Invoke PNET service routines
IPW$IRY	Invoke queue/account file recovery
IPW$IXS	Invoke Spool access support subroutines
IPW$OEF	Open diskette file
IPW$OPI	Invoke user ouput parameter processing routine
IPW$OTP	Open/close tape
IPW$SRJ	Scan reader JECL statement
IPW$SSJ	Invoke parameter checker
IPW$SXJ	Scan execution JECL statement
IPW$ULP	Update LUB and PUB tables
IPWPUT	Invoke PNET Composer IPW$$NC (not used)

Figure 10 (Page 1 of 2). Service Macros

Macro	Purpose
Task management	
IPW$ATT	Attach new task
IPW$DET	Detach current task
IPW$WFB	Wait for BSC event
IPW$WFC	Wait for single posting
IPW$WFD	Wait for dispatch
IPW$WFE	Wait for single posting
IPW$WFI	Wait for initiation
IPW$WFL	Wait for locked resource
IPW$WFM	Wait for multiple posting
IPW$WFO	Wait for operator
IPW$WFQ	Wait for class table posting
IPW$WFS	Wait for storage posting
IPW$WFX	Wait for mixed ECB and class table posting
Resource management	
IPW$RLR	Release resource
IPW$RSR	Reserve resource
Storage management	
IPW$RLV	Release virtual work space
IPW$RLW	Release real work space
IPW$RSV	Reserve virtual work space
IPW$RSW	Reserve real work space
IPW$UNV	Unchain virtual work space
Message service	
IPW$GAM	Get message text
IPW$GTR	Get message (no longer used)
IPW$ICS	Nodal message service
IPW$NTY	Notify message service
IPW$RMS	Remote message service
IPW$WTO	Write to operator
IPW$WTR	Write to operator with reply
Disk service	
IPW$WTQ	Write queue record block or master record
IPW$RDQ	Read queue record block or master record
IPW$WTD	Write data block
IPW$RDD	Read data block
Tape service	
IPW$WTT	Write tape record
IPW$RDT	Read tape record
IPW$CTT	Execute tape control
Timer service	
IPW$RDC	Read (TOD) clock
IPW$STM	Set timer interval
Validation service	
IPW$VDA	Validate data area addresses
Remote service	
IPW$SRM	Set remote mask in bit table
Trace service	
IPW$GTE	Get trace entry
Queue File service	
IPW$GQR	Get queue record
IPW$MQR	Modify queue record
IPW$WQR	Write queue record

Figure 10 (Page 2 of 2). Service Macros

Macro	Purpose
Multiprocessor Service	
IPW$TDM	Switch NP/PU Mode

Figure 11 (Page 1 of 2). Definition Macros

Macro	Purpose
IPW$DAB	Define asynchronous service anchor block (ACB)
IPW$DAC	Define account control block (ACB)
IPW$DBA	Define virtual buffer control area
IPW$DCB	Define command control block
IPW$DCI	Define communicator information block
IPW$DCM	Define RJE, BSC commands
IPW$DCO	Define compaction table control block (COCB)
IPW$DCP	Define control blocks used by command proc.
IPW$DCT	Define class table entry
IPW$DCW	Define channel command word
IPW$DDE	Define device entry
IPW$DDR	Define data record format
IPW$DED	Define external device control block
IPW$DEF	Define general use control blocks
IPW$DFC	Define printer control record
IPW$DGN	Define generation table (GNB)
IPW$DJK	Define layout of account file control interval (HEADER, CDIF, RDF)
IPW$DKA	Define PNET compression/decompression work area
IPW$DLC	Define line control block (LCB)
IPW$DLR	Define logon request control block (LRCB)
IPW$DLU	Define logical unit control block (LUCB)
IPW$DLW	Define logical reader work area
IPW$DMD	Define message
IPW$DMC	Define module control block (MCB)
IPW$DMM	Define message control block (MSCB)
IPW$DMS	Define RJE (BSC and SNA) message control block
IPW$DNC	Define node control block
IPW$DNR	Define network control records
IPW$DOP	Define output parameter interface block
IPW$DPA	Define permanent area (CAT)
IPW$DPD	Define partition control block (PDB)
IPW$DPN	Define PNET master control block
IPW$DPW	Define physical work space (PWS)
IPW$DQC	Define disk management block (DMB)
IPW$DQR	Define queue record (QRA)
IPW$DRM	Define SNA remote control block (RMCB)
IPW$DRQ	Define PNET SNA session request block
IPW$DSC	Define storage control block (SCB)
IPW$DSD	Define storage descriptor
IPW$DSL	Define SLI work space (SLW)
IPW$DSN	Define SNA control block (SNCB)
IPW$DSP	Define spool environment header and record & block
IPW$DSR	Define service request block (SRB)
IPW$DSS	Define PNET SNA session control block
IPW$DSU	Define SNA unit control block (SUCB)

Figure 11 (Page 2 of 2). Definition Macros

Macro	Purpose
IPW$DSV	Define save area
IPW$DTB	Define tape control block (TBB)
IPW$DTC	Define task control block (TCB)
IPW$DTE	Define task control block extension area
IPW$DTI	Define RJE,BSC task identifiers
IPW$DTP	Define TCP/IP Driver Control Block (TDCB and SDCB)
IPW$DTX	Define transmitter exit parameter list
IPW$DVC	Define PNET VTAM control block
IPW$DVD	Define various DSECTS
IPW$DVP	Define various PNET DSECTS
IPW$DVS	Define virtual storage control block
IPW$DWA	Define SNA work area (WACB)
IPW$DWC	Define PNET composer work area
IPW$DWG	Define PNET receiver/transmitter work area
IPW$DWN	Define receiver/transmitter account area
IPW$DWP	Define PNET presentation service work area
IPW$DXE	Define output exit parameter list
IPW$DXW	Define cross-partition work area
IPW$IOR	Define input/output request (RJE,BSC)
IPW$MXD	Define segment macro IPWSEGM work area

Figure 12. Miscellaneous Macros

Macro	Purpose
IPW$AJ#	Assign new VSE/POWER job number
IPW$ALN	Align to storage boundary
IPW$CPY	Provide copyright instruction
IPW$VCA	Validate command authority
IPW$EQU	Establish equates
IPW$GMD	Generate message definition
IPW$GMM	Generate message module
IPW$RET	Restore registers and return to caller
IPW$SAV	Save caller registers

Figure 13. PNET TCP TD-Subtask Support

Macro	Purpose
IPW$GTO MSG=	Issue message for TD-Subtask
IPW$GTO TRACE=	Issue trace message for TD-Subtask
IPW$GTO DOM=	Delete message for TD-Subtask
IPW$ITP PARMS=	EZASMI API socketcall
IPW$ITP CKRC=YES	EZASMI API socketcall error checking
IPW$TTM STXIT=YES	Initialize STXIT interface for TD-Subtask
IPW$TTM TIME=(Rx),TQE=	Specify timer interval interrupt
IPW$TTM CANCEL=YES,TQE=	Cancel timer interval interrupt
IPW$TTM PROCESS=YES	Process timer interval interrupt(s)
IPW$TTM WAIT=(Rx)	Set TD-Subtask in wait for Rx interval
IPW$TTM WAIT=(Rx,REACTIVATE)	Set TD-Subtask in wait for Rx interval, and then reactivate timer interface

Figure 14. PNET SSL SD-Subtask Support

Macro	Purpose
IPW$GTS MSG=	Issue message for SD-Subtask
IPW$GTS TRACE=	Issue trace message for SD-Subtask
IPW$GTS DOM=	Delete message for SD-Subtask
IPW$ITS PARMS=	EZASMI API socketcall
IPW$ITS CKRC=YES	EZASMI API socketcall error checking
IPW$TTS STXIT=YES	Initialize STXIT interface for SD-Subtask
IPW$TTS TIME=(Rx),TQE=	Specify timer interval interrupt
IPW$TTS CANCEL=YES,TQE=	Cancel timer interval interrupt
IPW$TTS PROCESS=YES	Process timer interval interrupt(s)
IPW$TTS WAIT=(Rx)	Set SD-Subtask in wait for Rx interval
IPW$TTS WAIT=(Rx,REACTIVATE)	Set SD-Subtask in wait for Rx interval, and then reactivate timer interface

Initialization and Termination

Initialization of VSE/POWER

The initialization of VSE/POWER comprises of the following phases:

- User-generated phase (POWER/IPWPOWER/username)
- IPW$$IP
- IPW$$I1
- IPW$$I2
- IPW$$I3
- IPW$$I4
- IPW$$I5 (optional for Accounting support)
- IPW$$I7
- IPW$$IN (optional for VSE/POWER PNET support,
  refer to "PNET Initialization").
- IPW$$T1 (entered at end of initialization, awaiting reactivation by PEND
  command, refer to "Termination of VSE/POWER")

Job control (EXEC statement processor) fetches the first of these phases, which contains a small loader routine and a generation table. These are assembled from the generation macros POWER, PLINE (optional) and PRMT (optional). There can be as many of these generation table phases in the library as there are different versions of VSE/POWER needed by the user.

The loader routine in front of the generation table loads the initialization root phase behind the first page in the pageable area (the first page is reserved for the permanent command processor task as work area) and gives control to it (see Figure 15 on page 40).

In case the pageable area is not large enough to contain the initialization root phase, a message is issued by the generation-table load routine and the initiation of VSE/POWER is canceled.

The root phase loads all necessary initialization phases one after the other in the overlay area that is part of the root phase and passes control to them. After processing of an overlay phase, control is given back to the root phase, which then loads the next overlay phase. This process continues until the termination phase (IPW$$T1) is loaded.

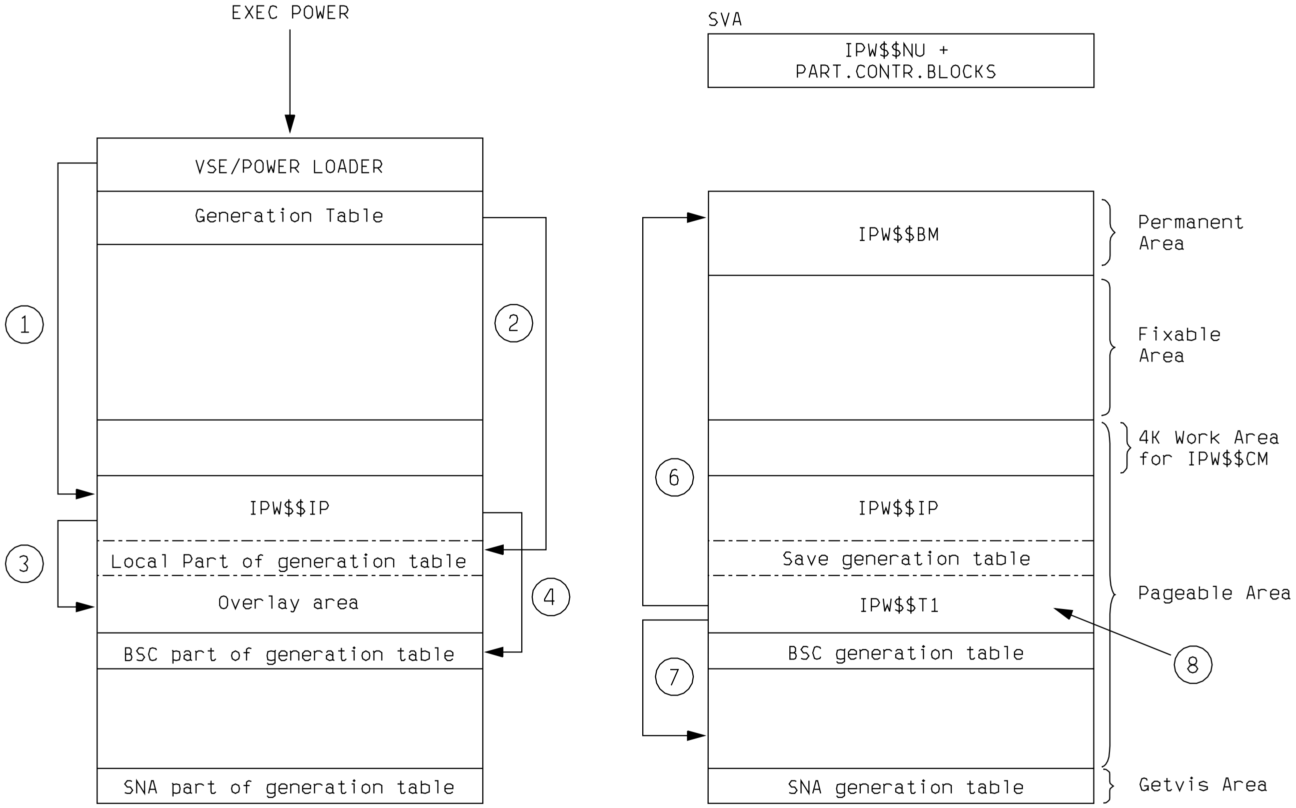


Figure 15. Initiation Logic

Notes:

1. VSE/POWER Loader loads IPW$$IP (start of pageable area + 4K).
2. IPW$$IP saves the local part of the generation table internally.
3. IPW$$IP loads the initialization phases into the overlay area and gives control to them.
4. IPW$$I1 saves the BSC portion of the generation table behind IPW$$IP.
5. IPW$$I1 builds the SNA control block (if required) in the GETVIS area and saves the SNA portion of the generation table.
6. IPW$$I1 loads IPW$$NU into the SVA and IPW$$BM (if BSC is supported) into the partition and fixes them.
7. The other VSE/POWER phases are loaded into the pageable area.
8. The last phase loaded by IPW$$IP is IPW$$T1; this is the termination phase of VSE/POWER.

IPW$$IP contains the following services and DTFs:

- IPLOAD loads VSE/POWER phases and exits.
- IDA00 builds IDAL lists for MCBs.
- FM00 formats queue file and data file MCBs.
- FD00 formats the data file during COLD start.
- FX00 formats one data file extent during extension warm start and is called via FD00 (R0 ≠ 0 &rpar) .
- PS00 prints a Status Report if at warm start SYSLST is assigned to a printer or at PEND time when a printer address is given.
- SD00 sets up the DTFPH for each file.
- DTFPH for the following files:
  - IJQFILE for the queue file.
  - IJAFILE for the account file.
  - IJDFILE for the data file.
  - IJSYSIN for the input file.

  - IJDTEST for the test data file extent file.
  - IJQFILE for the queue file.

IPW$$I1:

- Checks if VSE/POWER is already active.
- Checks if VSE/POWER runs as main task.
- Checks if VSE/POWER runs in a virtual partition.
- Issues SVC 13 to change PSW key to zero, and hense continues in NP-mode, provided Turbo Dispatcher is active.
- Checks if the VSE/AF Supervisor supports JAI, if VSE/POWER accounting is requested.
- Checks if real storage allocated to VSE/POWER partition is large enough.
- Checks if RJE,SNA is supported.
- Validates the remote work station ids for RJE,SNA.
- Builds the remote control block (SNA).
- Saves the SNA portion of the remote work station table in GETVIS area.
- Checks if RJE,BSC is supported.
- Validates line address(es) for RJE,BSC.
- Saves the BSC portion of the line/remote work station table behind IPW$$IP.
- Informs other subsystems that VSE/POWER is active.
- Loads VSE/POWER nucleus into pfixed system GETVIS area (SVA)
- Loads RJE,BSC monitor (if applicable) and issues PFIX command for it.
- Initializes storage control block (SCB) and storage fields in CAT.
- Obtains DOS/VS related information.
- Initializes local message control block (MMB).
- Relocates various addresses.
- Determines pageable area required depending on required functions.
- Loads all applicable command processor phases.
- Loads all VSE/POWER modules in the pageable area.
- Allocates trace area in pageable storage, if applicable.
- Loads local user reader exit, if applicable.
- Loads job exit, if applicable.
- Loads local user output exit, if applicable.
- stores exit data in the initialization processor work area

IPW$$I2:

- Defines cross-partition ECBs for spool management.
- Creates TCBs for initiation/termination task and permanent command processor task.
- Sets up virtual storage control block (VSCB).
- Builds the initial TCB chain, consisting of the wait control block, the command processor task control block, and the initialization processor task control block.
- Calls 'IPW$TDM NP', so that the non-parallel mode is also reflected in TCF16 of Init-Task, and enforces default NP-processing by setting CAF4WKNP=ON - being prepared for change by SET WORKUNIT=PA.
- Establishes AB-Exit routine.
- Indicates in SYSCOM that VSE/POWER is active
- Extracts CPUID and saves it in IP-workarea
- Initializes 16 access registers with zero and obtains the access list entry token (ALET) of the VSE/POWER partition if running in ESA mode on ESA hardware.
- Establishes linkage to operator communication routine.
- Establishes linkage to interval timer exit routine.
- Checks if VSE/POWER runs in shared partition (/370 mode only).
- Sets up trace information block (TIB) and attaches DUMP subtask, if applicable.
- Attaches asynchronous service subtask

- Saves session start date and time.
- Checks if running with new device structure (>254 devices).
- Checks if SYSIPT is assigned and if so, reads first autostart statement and saves it for later PDISPLAY AUSTMT
- Processes SET autostart statements, if any.
- Processes DEFINE autostart statements, if any.
- Analyzes autostart statement (FORMAT=).
- Checks if supervisor is generated with shared DASD feature.
- Checks when queue file and data file are shared that the device(s) are defined as shared ones.
- Requires confirmation from the operator when cold start of one of the files was requested when running shared.
- Locks the queue file for exclusive use when running shared.
- Establishes linkage to temporarily used timer routine to issue message 1QB6I if applicable.
- Allocates the exit data table and sets up address in CAT.
- Allocates the FCB table and sets up address in CAT.

IPW$$I3:

- Obtains device characteristics (device type) of VSE/POWER queue file.
- Calculates number of queue record blocks per track.
- Reserves storage for and initializes queue-file MCB.
- Saves device characteristics in MCB.
- Sets up sector table in MCB (if applicable).
- Opens queue file and allocates real storage for queue record block input/output area.
- If OPEN fails IPW$$AT passes control back to IPW$$I3, which starts Q-File Re-Alloaction:
  - Reads and checks IJQFOLD DLBL/EXTENT using LABEL macro
  - Obtains old Q-File device characteristics
  - Calculates number of QRBs per track
  - Reserves storage/formats old Q-File MCB
  - Saves device characteristics in MCB
  - Sets up sector table in MCB
  - For Shared Spooling: Lock IJQFOLD disk if file does not reside on same disk assigned to SYS001
  - Opens old queue file IJQFOLD on SYS034
  - Reserves storage and formats DMB IJQFOLD
  - Sets up defaults and user values in DMB
  - Calculates total number of usable QRBs
  - Reads IJQFOLD Master Record
  - For Shared Spooling: Check for other system up and running
  - Checks IJQFILE DLBL/EXTENT
  - Checks that IJQFILE & IJQFOLD extents do not overlap
  - Asks operator to confirm queue file re-allocation
  - Adds IJQTEST DLBL/EXTENT with same location as IJQFILE
  - Verifies IJQFILE extent by OPEN of IJQTEST
  - Checks that IJQFOLD fits into IJQFILE by comparing the number of QRBs
  - Reserves IO-area for IJQTEST/IJQFILE
  - Reserves and formats DMB for IJQTEST (later used for IJQFILE)
  - Calculates total number of usable QRBs for IJQTEST
  - Reserves storage for incore copy of IJQTEST (later IJQFILE)
  - Formats new IJQTEST extent on disk
  - Returns to main routine addressing IJQFOLD as queue file (Return address is queue file mis-match check)
- Acquires storage for and formats disk management block (DMB).
- Sets up default and user-specified values (out of generation table) in DMB.
- Calculates total number of usable queue records.

- Acquires VIO or partition GETVIS-31 space for storage copy of queue file.
- Sets up record-control fields in DMB.
- >>> Queue file Re-Allocation routine returns here after successful 1st part
- Checks for mismatch of queue file and offers "Release Migration During Warm Start" for a queue file >= 6.7.
- Sets up default and standard values in DMB from generation table.
- Sets up VSE security SECNODE value based on cold/warmstart
- Sets up shared refresh bit mask in DMB.
- Calculates queue file limit percentage.
- Formats queue file and builds free queue record chain, if cold start.
- Determines type of start (warm start or abnormal warm start) including detection of equal shared SYSID by means of CPUID.
- Reads in all queue record blocks if applicable
- For Q-File Re-Alloaction the old queue file is now extended in storage and written to the new location.
  - IJQFOLD DMB is copied to the IJQTEST DMB
  - The IO-words for the queue file and the master record are rebuild
  - The queue file limit is recalculated
  - The additional FREE queue records are appended if IJQFILE > IJQFOLD
  - The additional FREE queue records are appended if IJQFILE > IJQFOLD
  - All queue records and the Master Record are written to IJQTEST
  - IJQTEST is closed and re-opened (for output) as IJQFILE
  - IJQFOLD is overwritten by a DUMMY file
  - For Shared Spooling: IJQFOLD disk is UNLOCKed if its separate to the IJQFILE disk.

IPW$$I4:

- Obtains device information of device containing data file.
- Checks DBLK specification.
- Reads IJDFILE DLBL/EXTENT information from Label Area.
- Checks whether number of extents in DLBL/EXTENT exceeds number of extents of previous session by ONE or MORE. Then Data file Extension path is entered, which
  - Requests confirmation for assumed data file extension.
  - Assures that no other shared spooling system is logged on.
  - Compares DLBL/EXTENT information with extent information saved from previous session in Master Record field MRDFEXT (Logical Unit, start of extent, length of extent).
  - Checks that device information of all logical units is consistent (device type, shared or not)
  - Verifies planned location of additional extent(s) by OPEN test file.
  - Switches IJDFILE DTF to open for output (COLD start) if tests are successful.
  - Else DLBL/EXTENT truncated to previous EXTENT(s) is written into label area and normal warm start is continued.
- Opens data file, builds and formats one MCB for each extent.
- Acquires input/output area in the length of DBLK for each extent.
- Checks if more than one extent exists on same volume.
- Checks if the same number of data-file extents are specified as at cold start time.
- Checks when queue file and data file are shared that further data file extents which do not reside on the same volume as the 1st extent are on shared device(s).
- Builds sector table in MCB of first extent.
- Checks for matching DBLK and DBLK group size values in master record and on warm started data file.
- Formats the data file, if cold start, and builds the free DBLK group subchains.
- For extension warm start formatting of the appended extent(s) is delegated to IPW$$T1.
- Calculates data file limit percentage.

IPW$$I5:

- Obtains device information of device containing account file.
- Opens account file.
- Builds and formats ACB.
- Acquires input/output area (4K), if account file resides on FBA device.
- If cold start, erases account file.
- If warm start, locates last written account record or CI if FBA device by invoking account file recovery.

IPW$$IN:

Entered when PNET= is specified at VSE/POWER generation time -- see “PNET Initialization” on page 188 for more details.

IPW$$I7:

- Invokes queue-file recovery, if necessary (warm start).
- Updates in the master record, the maximum queue records used in present session, use count and the SYSID/CPUID table (shared only).
- Attaches time event scheduling task (TES) and waits until posted by TES task and writes then modified queue record blocks back to disk.
- Prints status report, if warm start and SYSLST assigned.
- Writes master record back.
- Attaches librarian subtask, if applicable.
- Sets up timer task, if queue file and data file are shared and waits until initialization task gets posted by timer task, so that later the “startup” account record is written within a T1 interval.
- Activates channel end appendage.
- Sets up line manager task (if RJE,BSC)
- Sets up dynamic partition control block (DPCB) and attaches dynamic partition scheduling task (DPST) if running in /370 or ESA mode.
- Builds short on real storage cushion.
- Builds master external device control block.
- Sets up spool access service master task.
- Sets up SNA control block (if RJE,SNA).
- Builds and formats remote message control block.
- Sets up spool management master TCB.
- Informs software inventory control about default SLI member type.
- Obtains system-related boundary information and saves it in CAT.
- Writes VSE/POWER “startup” account record.
- Performs autostart and informs system operator by message 1Q12I that VSE/POWER is up.
- Provides VSE/POWER 'up' information for GETFLD service.
- Activates page-fault appendage.
- informs system operator that VSE/POWER is up.

IPW$$T1:

Entered at end of initialization awaiting reactivation either for

- formatting additional data file extents as determined by IPW$$I4. After formatting, the data file limit percentage is re-calculated.
- scanning the DELetion queue for entries that need 'final freeing' via IPW$FQS LOCK=NO.
- PEND command entered by the operator -- see “Termination of VSE/POWER” on page 54 for more details.

Initial task selection by VSE/POWER is illustrated in Figure 16.

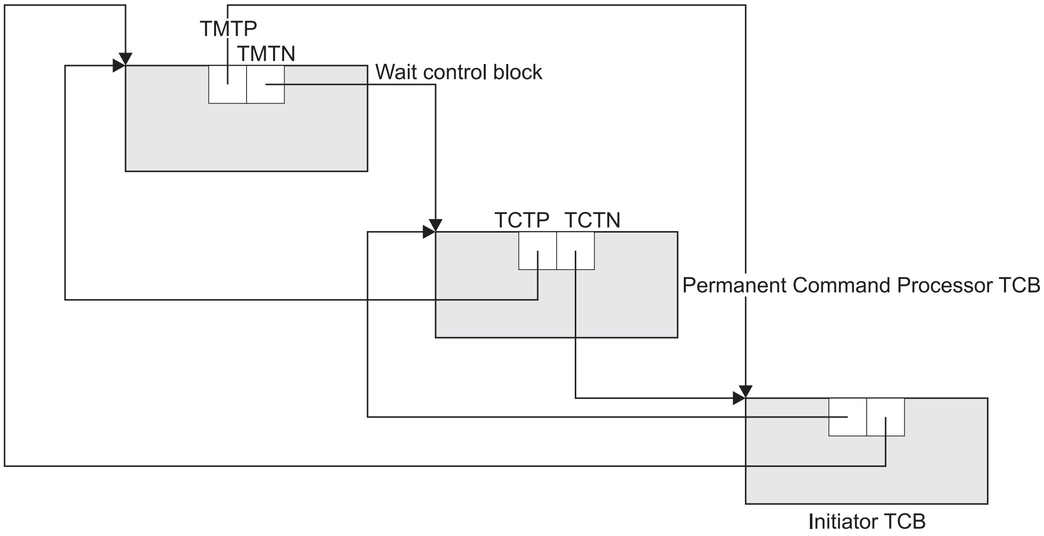


Figure 16. Initial Task Selection (TCB Chain)

VSE/POWER Startup

Currently there are three ways for a VSE/POWER system to be initialized:

- Cold Start
- Warm Start and Recovery Warm Start (partial/full recovery)
- Data File Extension Warm Start
- Queue File Re-Allocation Warm Start

Cold Start: Cold start can be performed individually for the Queue/Data and/or account file.
In a shared spooling environment, the operator is prompted via message 1QB2D for confirmation that no other VSE/POWER system is currently active. Depending on the reply, the initialization continues or terminates abnormally.
Cold start is performed by opening the appropriate file in 'write' mode.

- The queue file is initialized with queue record blocks. The size of a queue record block is 12288 bytes and contains 32 compartments. The compartment size is 384 bytes with each compartment containing one queue record of 368 bytes. If the queue file resides on a FBA device, a queue record block comprises 24 FBA blocks. The space of the last queue record block is used for the master record. Depending on the size of the queue file, 1 or 2 queue record blocks are occupied by the master record. All queue records are marked "free" and placed in the free queue record chain. The free queue record chain is shown in Figure 33 on page 90.

  Note: The first queue record (relative queue record number = 0) is reserved for internal purposes; it contains the queue record block number of the master record.

- The data file (CKD or FBA) is initialized with empty (hex zero) DBLKs. The DBLKs are grouped together in DBLK groups, that means every first DBLK in group is headed by a Spool Environment Header (SEH) record and every last DBLK in group is headed by a Spool Environment Record (SER). The free DBLK group subchains are built. The DBLK groups are distributed among 8 subchains, which are chained off the master record. The free DBLK groups are forward chained together via the SER, which contains the relative DBLK number of the first DBLK of the next DBLK group. The SEH record has no function in the free subchains.

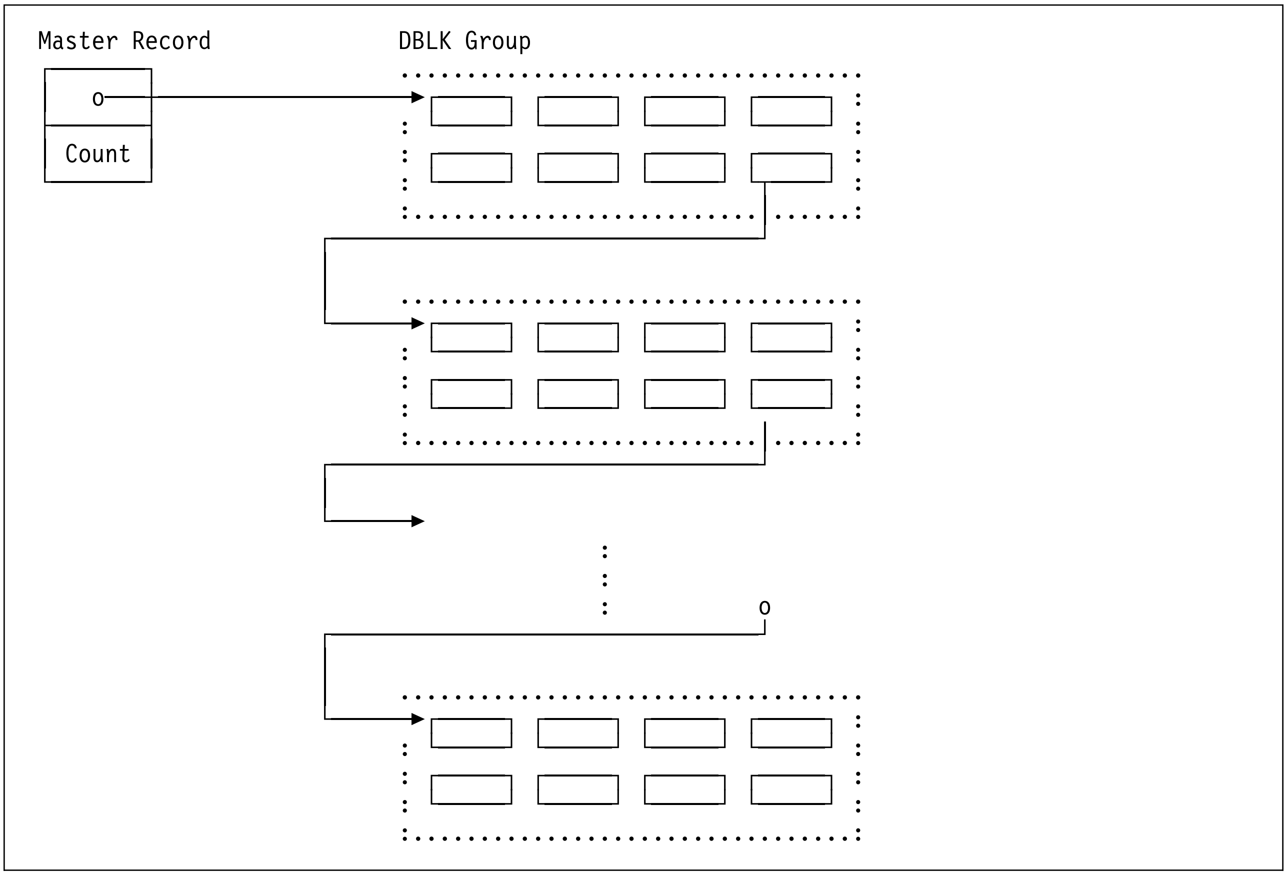


Figure 17. Free DBLK Group Subchain

For more detailed information on the logical structure of the free DBLK group subchains see also VSE/POWER Administration and Operation, Appendix B, 'Analyzing Dumps and Traces'.

- The account file is initialized with an EOF record written on each track when the account file resides on a C-K-D device. If the account file resides on an FBA device, an EOF-CI is written as first CI on the file.

Warm and Recovery Warm Start: The decision to perform a 'Warm Start' or a 'Recovery Warm Start' depends on the contents of a field (the USE count), contained in the master record. When initialization is complete the USE count is incremented by one, and when a VSE/POWER system terminates *normally* the USE count is decremented by one.

If at initialization time the USE count is zero, meaning that there are no VSE/POWER systems active and that the last VSE/POWER termination was *normal*, a 'Warm' start is performed. All information required to control the spool files are extracted from the master record, without re-constructing the class chains on disk.

If however the USE count is non-zero in a non-shared environment the queue file recovery program (IPW$$RY) is invoked to re-construct the class chains and to reset all queue entries which are still marked 'in execution'.

If the USE count is non-zero in a shared environment then another field, referred to as the SYSID/CPUID-bucket, is used to check if the last termination of this VSE/POWER system was normal or abnormal. If the bucket contains an entry with the same SYSID as the initializing system, then the last termination was abnormal. In this case 'recovery' warm start is performed. If however the last termination of this VSE/POWER system was normal, 'warm' start is initiated.

Note: Normal termination always resets the SYSID/CPUID-bucket for that system to zero if it is in a shared environment.

An additional field, contained in the master record, is the 'SYSID' field, that contains the SYSID of the system that is currently accessing the queue file in update mode. This field is important *after* VSE/POWER initialization, because whenever a VSE/POWER system locks the queue file, it first checks the SYSID field. If a SYSID is present then queue file recovery is performed for the SYSID found in the field, without the operator having to re-initialize VSE/POWER in the abnormally terminated system.

The scope of "recovery warm start" depends on the status of the queue file. VSE/POWER distinguishes between

- full recovery
- partial recovery (shared spooling only)

Full recovery is always performed in a non-shared spooling environment, when the last termination was abnormal or in a shared spooling environment when the master record indicates that either the initializing system or any other system abnormally terminated while having the queue file in "update mode" (within T1 interval).

Partial recovery is performed in a shared spooling environment when the last termination of the initializing system was abnormal, but the system was not 'owner' of the queue file (initializing system terminated outside of T1 interval).

Full Recovery: The various class chains and the free queue record chain are re-built by queue file recovery. Queue file recovery accesses (via IPW$GQR macro) each queue record in turn using the sequential organization (not dependent on chains) and the status of each queue record is examined. A queue record can be in one of the following states:

- 'free' (queue record is member of free queue record chain)
- incomplete & 'in creation' (queue entry is being created)
- queued 'inactive' in one of the various class chains
- queued 'active' (queue entry is in execution)
- incomplete & in 'delayed deletion' (queue entry is waiting for deletion until MACC is set to zero, for details see “Access to Active Queue Entry” on page 101)
- 'bad' - queue record is inaccessible due to I/O error

In details, following steps are performed: First the count of accessing SAS BROWSE tasks (MACC) is adjusted, by setting the count to zero for each recovered system. This enables correct handling of 'delayed deletion' entries as described later.

- If the queue record is marked 'free', the free queue record function is called to place the queue record back in the free chain.
- If the queue record is marked 'in creation', the queue record is placed in the free queue record chain and all allocated DBLK groups are returned to the free DBLK group chain, unless the queue entry represented by the queue record is checkpointed. In the latter case the queue entry is repositioned at the last checkpoint, a job trailer record is written as last record of the queue entry and the add to queue function is invoked by means of the IPW$AQS macro instruction to add the queue entry to the appropriate class chain.

  If, however, the queue record is 'in creation' on a system for which no recovery should be done, the queue record is left as is (shared spooling only).
- If the queue record is marked 'inactive', the add to queue function (IPW$$AQ) is called to add the queue entry into the appropriate class chain.
- If the queue record is marked 'active' by the recovering system or by the system that abnormally terminated (in T1), the execution flag is reset and the queue set is added to its class chain by invoking the

add queue function. If the queue record represents a RDR queue entry and the NORUN=YES AUTOSTART option was set, disposition 'X' is forced, too. If the queue record represents a LST or PUN queue entry and the "protect" option is set, the temporary disposition "Y" is set.

If the queue record is 'active' on the system for which no recovery is done, the queue record is added to the appropriate class chain but with the execution flag still set (shared spooling only).

- If the queue record is marked in 'delayed deletion', the free queue record function is called. If MACC is zero, the queue record is placed back in the free queue record chain and all allocated DBLK groups are returned to the free DBLK group . If MACC is not zero, meaning that the entry is still browsed, the queue entry stays in 'delayed deletion'.
- If the queue record is marked 'bad', no action is taken.

When running shared, all queue record blocks are written back to disk after recovery warm start is performed; the appropriate refresh bits are set to indicate other shared systems, that their queue record blocks in storage are obsolete and should be refreshed.

Partial Recovery: Partial recovery assumes that the queue file on disk with all its chaining pointers is valid. Only queue entries either marked 'in creation' or 'active' or in 'delayed deletion' for the initializing system or the system(s) to be recovered are reset. Again first the count of accessing SAS BROWSE tasks (MACC) is adjusted, by setting the count to zero for each recovered system. This enables correct handling of 'delayed deletion' entries as described later.

- If the queue record is marked 'in creation' by the initializing system, the free queue entry function is called to free the queue entry. If the queue entry was checkpointed, the queue entry is repositioned at the last checkpoint, a job trailer record is written as last record of the queue entry and the add queue function is invoked to add the queue entry to the appropriate class chain.
- If the queue record is marked 'active', the execution flag is reset and the queue record is written back to disk. If the queue record represents a RDR queue entry and the NORUN=YES AUTOSTART option was set, disposition 'X' is forced, too. If the queue record represents a LST or PUN queue entry and the “protect” option is set, the temporary disposition "Y" is set.
- If the queue record is marked in 'delayed deletion', the free queue record function is called. If MACC is zero, the queue record is placed back in the free queue record chain and all allocated DBLK groups are returned to the free DBLK group . If MACC is not zero, meaning that the entry is still browsed, the queue entry stays in 'delayed deletion'.

Data File Extension Warm Start The data file extension during warm start will not affect already spooled data and is triggered when VSE/POWER detects **ONE or MORE** extent(s) appended with ascending sequence number to the existing IJDFILE DLBL/EXTENT statements. The new extent(s) must be added as last extent(s), because VSE/POWER accesses the extents as a contigious stream of DBLKs, starting with DBLK #0 and ending with DBLK #n. DBLKs on existing extents are already referred by their number which can not be changed.
The data file extension will format only the new extent(s), which is postponed after warm start has been completed. While the additional extent(s) are formatted, spooling is no longer disabled as during formatting of queue and data file at cold start.
Leaving the already spooled data on the existing extents untouched and formatting the additional extent(s) in flight reduces System-down-time from several hours to the time needed for RE-IPL.

Now during warm start VSE/POWER analyses the DLBL IJDFILE information in the label area and compares the number of extents against the count saved from its last session.
If the new count is less the old count, VSE/POWER enters a warm start which terminates with message

```
1Q19I INVALID DATA FILE EXTENT, RC=0002
```

and messages 1Q0KI informing about IJDFILE DLBL and about data file used at previous session

```
1Q0KI DATA FILE EXTENT NO. 01 AS EXTRACTED FROM IJDFILE DLBL/EXTENT
      (// EXTENT SYSxxx,volid,1,0,start,length)
1Q0KI DATA FILE EXTENT NO. 02 AS EXTRACTED FROM IJDFILE DLBL/EXTENT
      (// EXTENT SYSyyy,volid,1,1,start,length)
- - -
1Q0KI DATA FILE EXTENT NO. 01 AS PRESERVED FROM PREVIOUS WARM START
      (// EXTENT SYSxxx,------,1,0,start,length)
- - -
```

If the new count is equal the old count, VSE/POWER performs a normal warm start.

If one or more extent(s) are appended, but the total number of extents exceeds the maximum of 32, VSE/POWER continues with a normal warm start and informs the operator by message

```
1QD1A TOO MANY ADDITIONAL EXTENTS (mm) FOR DATA FILE EXTENSION, RC=0002
```

If one or more extent(s) are appended, but the current total number of DBLKs is already the maximum of 2,147,483,647 VSE/POWER continues with a normal warm start and informs the operator by message

```
1QD1A TOO MANY ADDITIONAL EXTENTS (mm) FOR DATA FILE EXTENSION, RC=0003
```

If the new count exceeds the old count by one or more extents, VSE/POWER informs the operator by message

```
1QD7A mm ADDITIONAL EXTENT(S) FOUND FOR EXTENSION OF EXISTING DATA FILE WITH nn EXTENT(S)
```

and continues by first showing each already used extent as found in IJDFILE DLBL and EXTENTs by message

```
1QD2I EXISTING DATA FILE EXTENT NO. mm FOUND IN IJDFILE DLBL/EXTENT
(// EXTENT SYSxxx,volid,1,nnn,start,length)
```

and then asking the operator to confirm data file extension for **each** extent by message

```
1QD2D DATA FILE EXTENT NO. mm - FOR FORMATTING REPLY 'YES' ELSE 'NO'
(// EXTENT SYSxxx,volid,1,nnn,start,length)
```

If 'NO' VSE/POWER continues with warm start on the existing data file with message 1QD3A RC=000A regardless whether the 1st additional or a middle or the last additional extent has been rejected and regardless whether any extent has been confirmed before.
After answering 'YES' for each new EXTENT, VSE/POWER starts a process with the following stages.

1. For the already existing and for all appended extents VSE/POWER assures data file integrity by analysing and removing all BAM OPEN obstacles. The DLBL, EXTENT and ASSGN statements are inspected for obeying the rules of VSE/AF BAM and VSE/POWER, for example

   - All assignments (SYSxxx) must be in contiguous ascending order.
   - There must be no duplicate assignments to the same disk.
   - All extents must reside on disks of the same device type.
   - All existing extents are compared against the extent information contained in the Master Record, which presents the warm start information from disk and is part of the already opened queue file.

   If one of the preceding tests fails for a certain extent mm, VSE/POWER issues the warning message

   ```
   1QD3A DATA FILE EXTENSION FAILED FOR EXTENT NO. mm, RC=000x, WARM START CONTINUED
   ```

   and continues with a normal warm start on the existing data file.

   If the preceding tests are successful, a DLBL IJDTEST is added to the label area with the assignment and extent information of all **additional** extent(s), but a retention period of 0 days. Via OPEN for Output for IJDTEST VSE/POWER verifies whether the planned space is not occupied. The operator is informed by

   ```
   1QD4I VERIFYING LOCATION OF ADDITIONAL DATA FILE EXTENT(S) BY OPEN FOR 'IJDTEST'
   ```

   If the space is in use, message

```
4744D OVERLAP ON UNEXPRD FILE IJDTEST ... file id ...
```

will be shown, which should be replied by 'DELETE', if the unexpired file is no longer needed. In case the old file is still needed, VSE/POWER will switch to normal warm start on the existing data file after reply 'CANCELV' or 'DSPLYV' or ENTER. The operator is informed by message 1QD3A with RC=0007.

If the additional extents overlay each over, message

```
4740D EXTENT OVERLAPS ANOTHER  IJDFILE  SYS0xy=cuu  volume
```

and the verification fails. VSE/POWER will switch to normal warm start on the existing data file after ENTER. The operator is informed by message 1QD3A with RC=0007.

2. Step two starts when message

   ```
   1QD5I LOCATION OF ADDITIONAL DATA FILE EXTENT(S) VERIFIED SUCCESSFULLY
   ```

   has been issued. Then an OPEN for output for IJDFILE will rewrite the format-1/-3 labels of the existing extents and create the format-1/-3 label of the additional extent(s) in the corresponding VTOCs. Then VSE/POWER startup continues with EXTENT exit processing which records the new extent(s) in the master record with the "format additional extent" flag and posts the VSE/POWER formatting task. The old extents are **NOT** flagged for formatting thereby maintaining their spooled data.

3. When VSE/POWER initiation is complete (1Q12I), the third step starts indicated by highlighted message

   ```
   1QD6I FORMATTING OF NEW DATA FILE EXTENT NO. mm STARTED
   ```

   The named new extent is now formatted while VSE/POWER is spooling. When formatting has been completed for this extent, the operator is informed by message

   ```
   1QD6I FORMATTING OF NEW DATA FILE EXTENT NO. mm COMPLETED,
         nnnnnn FREE DBLKGPS ADDED
   ```

   The DBLK groups of the new extent are now chained to the existing free DBLKGP chain and are available for future spooling. The total number of available DBLKGPs (MRDBMAX) and the number of free DBLKGPs are adjusted at that moment.
   This 3rd step is repeated for each new extent.

Error Recovery: If VSE/POWER finds an incorrect IJDFILE EXTENT it issues message 1QD1A or 1QD3A with a unique reason code to inform the operator. To enable VSE/POWER warm start and to protect the original queue file and data file, it then truncates the IJDFILE label to the number of extents used by the previous session and passes the modified label to the temporary partition label area. The modified label should represent the previously used data file. However, if the system administrator has exchanged the order of the existing extents, VSE/AF BAM OPEN assures that mismatches between the modified label and the file labels on disk result in an OPEN error.

If VSE/POWER is terminated abnormally before the new format-1/-3 label has been written into the VTOC, the number of data file extents is not yet updated in the master record. A subsequent VSE/POWER start (with the same DLBL/EXTENT/ASSGN information) will detect the additional extent again and data file extension processing starts from the beginning.

If VSE/POWER is terminated abnormally while formatting the new extent(s), a subsequent VSE/POWER start will detect the additional extent(s), which are still regarded as unformatted ("format additional extent" flag is still set) and formatting is resumed automatically.

If VSE/POWER is terminated abnormally while formatting of new extent(s) is still incomplete and if VSE/POWER is started subsequently with even more extent(s) added to expand the data file once more, extension will be rejected with message

```
1QD1A TOO MANY ADDITIONAL EXTENTS (mm) FOR DATA FILE EXTENSION, RC=0004
```

and formatting of the previously added extent(s) is resumed.

If VSE/POWER is terminated by PEND while formatting of an additional extent is in process but still incomplete, termination is postponed until formatting of the currently addressed extent is complete. The possibly residual - not yet formatted - extents will be formatted after the next warm start. The operator is informed by message

```
1QD6I FORMATTING OF NEW DATA FILE EXTENT NO. mm POSTPONED TILL NEXT WARM START
```

If formatting of one additional extent fails due to I/O errors, VSE/POWER continues with the old extents plus the new extents formatted so far. The operator is informed by message

```
1QD6I FORMATTING OF NEW DATA FILE EXTENT NO. mm FAILED, RC=0003/0004
```

The system administrator should either remove the failing EXTENT and its successors or he should replace the failing extent and may at the same time also replace or remove any succeeding extent. At the next VSE/POWER warm start data file extension will be started for the new additional extent(s).

Considerations for Shared Spooling: If VSE/POWER shares queue and data file with other systems (SHARED=Q|D), special considerations ensure data integrity. After confirming Dynamic Data File Extension ('YES' replied to 1QD2D for all extents), it is checked, that no other system is already initialized. If the master record indicates another initialized shared spooling system, the operator is informed by message

```
1QD3A DATA FILE EXTENSION FAILED FOR EXTENT NO. mm, RC=0008 , WARM START CONTINUED
```

that VSE/POWER has switched to a normal warm start preventing that DBLKs of the additional extent are added to the free DBLKGPs chain. This would allow access by other systems running without the additional extent information which would be identified as Data File corruption.
If **other** systems have terminated abnormally, the operator may use PRESET to reset their 'active' status. To check whether **other** systems are initialized, the operator may use the PDISPLAY STATUS report showing all active systems, especially the 'others' e.g. 4,3,1

```
F1 0001 1R46I  TIME INTERVALS FOR SHARED SPOOLING (SYSID=2) :
F1 0001    T1 =   5 SEC., T2 =   0 SEC., T3 =  60 SEC., T4 = 180 SEC.
F1 0001    ACTIVE SYSID'S FOUND: 4,3,2,1
```

Also it is checked for shared spooling during step one whether all extents reside on shared disks. If not message 1QD3A with RC=0009 is issued and VSE/POWER continues with normal warm start. To distinguish which system must format the new extent and to allow another system to take over formatting if the first system fails and can not recover, a "formatting SYSID" will now be recorded in the master record together with the "format additional extent" indication. If during warm start a shared spooling system finds the "format additional extent" flag set together with the formatting SYSID of another system, highlighted informational message 1QD6I is issued

```
1QD6I FORMATTING OF NEW DATA FILE EXTENT NO. mm DETECTED ON SYSID sysid
```

The operator may then check whether the formatting system (named by sysid) is still alive. If it has terminated, a restart of the formatting system envokes queue file recovery, which will resume formatting. If that system can't be restarted, the PRESET command - addressing the failing system by its sysid - can be used to transfer formatting of the residual unformatted extents to the active system, which issued the PRESET.
Note, that Queue file recovery may also be entered automatically (with the same result) when one system detects, that the formatting system has terminated abnormally.

Queue File Re-allocation (without Re-Formatting the data file) at warm start: Re-allocation means, that the queue file is placed at a different location and that it may be extended at the same moment. The queue file re-allocation does not affect already spooled queue entries and adds the additional queue records of the increased queue file as free queue records. This leaves the linkage between existing queue entries and the data file as is, and therefore the data file must **not** be formatted and system-down-time is reduced visibly to the time needed for re-IPL and for formatting and copying the queue file to the new location. The new function is triggered when VSE/POWER detects **new** DLBL/EXTENT statements for IJQFILE addressing a disk area assigned by **SYS001** and named with a 'file-id' not yet listed in the appropriate VTOC. The previously used queue file must address its original disk area by DLBL/EXTENT statements for IJQFOLD assigned to **SYS034** with its original 'file-id'. (Note: The latter used to be identified during the previous sessions as IJQFILE assigned by SYS001.)

Note: To re-allocate the VSE/POWER queue file to the same disk you must use a different 'file-id' than the one of IJQFOLD which is already listed in VTOC. Otherwise VSE/POWER will warn the operator by message

1QE8A IJQFILE (// EXTENT SYS001,volid,1,n,start,length) MISMATCH WITH file-id

If you want to re-allocate the VSE/POWER queue file on its current disk with the same 'file-id', then you must first re-allocate the queue file to a second disk using the original 'file-id' and then you may re-allocate it to the 1st disk (again with the original 'file-id').

Then during warm start, VSE/POWER tries to open the not yet existing IJQFILE. VSE/AF handles the OPEN request by locating the 'file-id' in the VTOC of the disk assigned to SYS001 and if this fails, this is indicated by VSE/AF message

```
4601I NO FORMAT 1 LABEL FOUND  IJQFILE  SYS001=cuu  volid
```

Then VSE/POWER informs the operator by message

```
1QE1I  RE-ALLOCATION PROCESS STARTED FOR VSE/POWER QUEUE FILE
```

and tries to open the existing (old) queue file IJQFOLD assigned to SYS034. If this fails due to incorrect or missing DLBL/EXTENT/ASSGN statements for IJQFOLD, the operator is informed by message (RC=0020 or 0021)

```
1QE2A  RE-ALLOCATION OF QUEUE FILE FAILED, RC=nnnn. WARM START TERMINATED
```

When the existing queue file is also not found, VSE/POWER is canceled by VSE/AF with message

```
4601I NO FORMAT-1 LABEL FOUND IJQFOLD SYS034=cuu volid
```

After successful open of the previously used queue file (IJQFOLD), VSE/POWER checks first whether the old queue file DLBL/EXTENT addresses the correct location, otherwise the operator is informed by message

```
1QE2A  RE-ALLOCATION OF QUEUE FILE FAILED, RC=0005. WARM START CONTINUED FOR IJQFOLD ON SYS034
```

The DLBL/EXTENT for IJQFILE is analysed, if the DLBL can not be read, the operator is informed by message 1QE2A (RC=000B) and warm start continues with IJQFOLD. When the new queue file (IJQFILE) does overlap the extent of the old queue file (IJQFOLD), the operator is informed by message 1QE2A (RC=0003) and warm start continues with IJQFOLD. When IJQFILE and IJQFOLD reside on two disks with the **same VOLID** but with **different cuu**, the operator is informed by message 1QE2A (RC=000A) and warm start continues with IJQFOLD. When the new queue file can be accepted, the operator is informed about both (old and new) queue file extents and asked to confirm re-allocation by

```
1QE3I  IJQFOLD: // EXTENT SYS034,volid,1,n,start,length
1QE3I  IJQFILE: // EXTENT SYS001,volid,1,n,start,length
1QE3D  CONFIRM QUEUE FILE RE-ALLOCATION FROM IJQFOLD TO IJQFILE BY 'YES' ELSE 'NO'
```

For 'NO', message 1QE2A with RC=0009 is issued and warm start for IJQFOLD is continued, for 'YES' VSE/POWER will process the old queue file and the new queue file in parallel:
1) The operator is informed by message 1QE4I that the extent of the new queue file is now verified for being unused and that therefore a test queue file is opened on the same location.

```
1QE4I VERIFYING LOCATION OF NEW QUEUE IJQFILE FILE BY OPEN FOR 'IJQTEST'
```

This test file IJQTEST uses the EXTENT information of IJQFILE, but is defined with a retention period of zero days and with a different 'file-id'. When later re-allocation is completed, the VTOC entry of the test file is changed to the 'file-id' and retention period of IJQFILE, thus changing the test file to the permanent queue file without affecting the already re-allocated queue records. If the space is in use, message

```
4744D OVERLAP ON UNEXPRD FILE IJQTEST SYS001=cuu volid 'file-id'
```

is shown by VSE/AF. In case the named file is still needed, the operator replies 'ENTER' or 'CANCELV' or 'DSPLYV' and VSE/POWER switches to normal warm start on the old queue file and issues message 1QE2A with RC=0006. If the unexpired file is not needed, the operator replies 'DELETE'. After deleting the old file(s) or when no unexpired file was found at all, the following message is issued:

```
1QE5I LOCATION OF NEW QUEUE FILE IJQFILE VERIFIED SUCCESSFULLY
```

When the size of new queue file (IJQFILE) is compared against the size of the old one. If the new one is smaller than the old queue file, the operator is informed by message 1QE2A (RC=0002) and warm start continues with IJQFOLD.
2) Now PFIXed storage is reserved for internal work areas, if reservation fails the operator is informed by message 1QE2A (RC=000C) and warm start continues with IJQFOLD. Then storage for the new larger queue file is reserved either in Partition Getvis of VSE/POWER (running in a non-shared address space) or in VIO, when the VSE/POWER partition resides in shared (ALLOC S,F1=xxM). If reservation fails the operator is informed by message 1QE2A (RC=0007) and the reservation is retried with the size needed for the previously used and still existing smaller old queue file.
3) When sufficient storage is obtained for the new queue file, its disk location is formatted into queue record blocks, which consumes nearly no time (compared with formatting a data file). Any I/O error during formatting terminates re-allocation indicated by message 1QE2A, RC=0008 and warm start for IJQFOLD is continued.
4) The storage copy of the old queue file - already read into the storage area of the new queue file - is extended by the free queue records of the larger new IJQFILE and is committed to disk. Any I/O error when writing the extended queue file to the new location is indicated by message 1QE2A, RC=000F or RC=0010 and warm start for IJQFOLD is continued.
5) Then the IJQTEST file is closed and reopened for output as IJQFILE, which updates the VTOC entry with the specified 'file-id' and retention period. If this fails the operator is informed by message 1QE2A (RC=000D or 000E) and warm start continues with IJQFOLD. Else re-allocation is done and the operator is informed by message

```
1QE6A RE-ALLOCATION FOR IJQFILE COMPLETED, nnnnn FREE QUEUE RECORDS ADDED
```

6) Finally the original old queue file is overwritten by a 'DUMMY' work file (without operator intervention needed). If this fails the the operator is informed by message

```
1QE7I DELETION OF IJQFOLD FAILED, REMOVE FILE-ID 'file-id' ON volid MANUALLY
```

that the system administrator should remove the now superfluous IJQFOLD VTOC entry.

Considerations for Shared Spooling: When VSE/POWER re-allocates the queue file, it requires exclusive access of the old and the new queue file. If VSE/POWER shares queue and data file with other systems (SHARED=Q|D), special considerations ensure queue file integrity. VSE/AF LOCK requests for 'IJQFL.volid' are issued for the new queue file and, if JQFOLD resides on a separate disk, also for the old queue file to ensure that only 1 VSE/POWER shared spooling system initializes at a time. The re-allocating system, having both files now locked, checks, that only itself has accessed the old queue file so far (remember that the new queue file does not exist yet). Otherwise the operator is informed by message 1QE2A with RC=0001 and warm start for IJQFOLD is continued.

Also for shared spooling VSE/POWER checks, whether the new queue file resides on a shared disk. If not, the operator is informed by message 1QE2A with RC=0004 and warm start for IJQFOLD is continued. Furthermore the old queue file is deleted after re-allocation, to prevent access by other shared spooling systems with outdated IJQFILE DLBL.

Invocation: To prepare a queue file re-allocation warm start of VSE/POWER, the system administrator has to modify the DLBL/EXTENT statement for IJQFILE in STDLABEL.PROC and the ASSGN statement for IJQFILE in DTRPOWR.PROC :

1. DLBL of the existing IJQFILE must be renamed to IJQFOLD, EXTENT & ASSGN must be changed from SYS001 to SYS034.
2. A new DLBL and EXTENT statement for IJQFILE must be added to STDLABEL.PROC with logical unit SYS001, and the matching ASSGN statement must be added to DTRPOWR.PROC. Note, that the 'file-id' of the old and the new queue file may be equal!
3. Either VSE/POWER partition Getvis (ALLOC F1=... statement in $0JCL procedure) or the VIO area (1st statement in $IPLESA procedure) must be increased. Note, that for each 32 queue records 12K more storage is needed to hold the storage copy of the new queue file.
4. The number of programmer logical units for the VSE/POWER partition is specified in the NPGR statement, which is part of $0JCL procedure. At least 35 logical units must be specified, the default of 30 logical units is not sufficient!
5. Re-allocating the queue file from CKD disk to FBA disk or vice versa is allowed, but needs a careful calculation of the present size and the new size. The same is needed when re-allocating the queue file on CKD disks with different CKD track sizes. For these queue file size calculations use the VSE/POWER Administration and Operation manual.

Restrictions: To re-allocate the queue file to the **same disk** different file-identifiers must be used for the old and the new queue file, because VTOC must only contain unique file-identifiers.

For a shared spooling complex, all systems must be terminated, before one system can re-allocate the queue file.

When re-allocation fails, VSE/POWER does not terminate but addresses the old queue file by logical by programmer logical unit SYS034 during that session until the next start of VSE/POWER has either been completed by a successful re-allocation, or the IJQFILE DLBL has been reset to its original value. For this session the VSE/POWER spool file dump tool IPW$$DD can not use the IJQFILE DLBL/EXTENT/ASSGN statements provided in STDLABEL.PROC and DTRPOWR.PROC, which addresses the new, not yet existing queue file. To use IPW$$DD, partition DLBL/EXTENT/ASSGN statements must be provided which address the old queue file by IJQFILE and SYS001, as done in the previous session.
OEM programs accessing the queue file by their own I/O must use the logical unit information from the queue file CCB which is part of the queue file MCB.
PDISPLAY Q informs about the logical unit of the currently used queue file.

Termination of VSE/POWER

VSE/POWER is normally terminated by the PEND command. All active tasks are allowed to continue until they finish processing the current queue entry. Deactivation is handled by each of the tasks, after the command processor (IPW$$CM) has set a termination code ("S", "E", "F", or "H") in their TCBs. In case of an I/O error or logic error VSE/POWER tasks can also be terminated by the IPW$$TR routine (see “Abnormal Termination of VSE/POWER Tasks” on page 55).

After all supported partitions have been released, the VSE/POWER partition is restored for normal VSE/AF operation.

The detach routine of task management actually gives control to the terminator routine IPW$$T1 by posting the termination task when no more tasks, except of command processor, SAS master task and timer task (shared spooling only) are active.

IPW$$T1:

Entered when initialization is complete, awaiting reactivation by PEND command.

- Checks and waits for not more than minimum number of VSE/POWER tasks active - reactivation by task management.
- Detaches if applicable the dump subtask.
- Detaches the Librarian subtask, if applicable.
- Detaches the TES task, if applicable.
- Writes the final VSE/POWER execution account record.
- Optionally prints status report, especially for the queue file, by passing an internal PDISPLAY command to invoke Print status task (if SYSLST assigned to a printer).
- Writes all queue record blocks back to disk and updates the master record accordingly.

  Note: In a shared spooling environment only modified queue record blocks are written back to disk.

- Decrements use count, resets SYSID/CPUID-bucket and writes back master record to disk if running non-shared or allows for timer task to do so if running shared.
- Deactivates the timer task if shared spooling was active.
- Writes end-of-file record to Account file, if applicable.
- Closes VSE/POWER files (Queue, data and Account file).
- Deactivates the spool access support master task, which in turn terminates the notify task, if present and the heartbeat task, if present.
- Issues termination message 1Q21I.
- Restores VSE/POWER partition for normal use.
- Issues the EOJ macro which also PFREEs the permanent area and all other fixed pages.

Note: Since VSE/POWER abnormal termination routine processing is established with OPTION=EARLY, IPW$$AT is also entered in case of EOJ termination. However no action is taken there, instead IPW$$AT is exited again by the EOJ macro.

Abnormal Termination of VSE/POWER Tasks: The task termination routine IPW$$TR is entered from task selection C state processing in case of an error at completion of any I/O operation, or if 'S' is posted in the TCB of a reader/writer task, or if the task encountered a severe logic error. It executes under the TCB of the failing task. The failing task is canceled.

The following specific failures necessitate VSE/POWER termination:

- Logic error of VSE/POWER function encountered
- I/O error while IPW$$TR is trying to recover

VSE/POWER can be terminated abnormally by the PEND FORCE or VSE/AF CANCEL command also.

Abnormal Termination Processing of VSE/POWER: The VSE/AF supervisor passes control to routine IPW$$AT when an abnormal termination condition for VSE/POWER or one of its subtasks is encountered.

In these cases VSE/AF stores the PSW and the registers at the time of abend in the abnormal-termination save area that is located at the entry of the routine (displacement X'14'). The stored PSW is not the original EC-mode PSW, but has been modified and contains the interruption code, the instruction length code, and the condition code. If a ESA supervisor is running, the original EC-mode PSW and the sixteen access registers are stored after the general registers.

When one of the VSE/POWER tasks itself detects an abnormal-termination condition, it issues the IPW$CNC (CANCEL) macro instruction, which stores the registers (but not the PSW) in the abnormal-termination save area and branches immediately to the abnormal-termination routine with cancel code X'FF'.

The first action of the abnormal-termination routine is to establish a VSE/AF lock 'ATGATE', so that the routine may be used serially by the VSE/POWER main task and its several VSE/AF subtasks.

Since the VSE/POWER Maintask may have entered AB-exit in Parallel (PU) or Non-Parallel (NP) Mode, and since AB-exit will update lowcore SYSCOM and COMREGs, NP mode is acquired by a local call for TDSERV FUNC=SWITCHNP (see 'AT250'). The system operator is informed about the abnormal termination of the VSE/POWER main task via message 1Q2CI containing the PSW, failing module name and module start address, if available. In case abnormal termination has been invoked by the IPW$CNC macro, the system operator is informed by message 1Q2DI. Then a formatted dump is created automatically calling the IDUMP in flight module IPW$$ID via macro IPW$IDM.
Only if the SET 1Q30D=YES autostart option has been used the operator is prompted via message 1Q30D to specify if he wants a dump of the VSE/POWER partition or not. If the operator replies with a wrong answer (neither 'YES' nor 'NO'), message 1Q30D is reissued. If the answer is 'YES', the operator is prompted again via message 1Q2ED whether or not a print-out of the storage copy of the queue file is wanted, provided it resides in the VIO area. If the operator replies with a valid printer or tape address, the VIO storage holding the queue file storage copy is printed on the specified device in SYSLST format. Finally a formatted dump is created calling the IDUMP in flight module IPW$$ID via macro IPW$IDM.

Thereafter all VSE/POWER controlled partitions are canceled with the 'DOCLEAN' request, VSE/POWER indicators in these partitions are turned off and all unit record assignments for spooled devices are released.

For a summary of the IDUMP sections and their contents see “Operation of module IPW$$ID” on page 162.

If an error, such as DUMP library full or not defined, occurs while writing the IDUMP to the DUMP library, VSE/POWER informs the operator via message 1QC5I about the cause of the error and prompts the operator via message 1QC5D to specify the printer or tape device on which he wants the IDUMP alternatively to be printed.

If he specifies an invalid device (such as no valid printer/tape device or device is down), the operator is prompted again. Otherwise SYSLST is assigned to the device just specified and the IDUMP is written in SYSLST format to the SYSLST device.

After the IDUMP has been taken which presents also VSE/POWER's partition control block for dynamic partitions from the system GETVIS area, VSE/POWER awaits end of 'CLEANUP' processing for all partitions under its control and requests then de-allocation of the dynamic partitions.

In all cases - dump requested YES or NO - the VSE/POWER partition itself will be tidied up. Indicators in the SYSCOM and the partition COMREG are reset to indicate that VSE/POWER is no longer active. All non DASD logical units are unassigned. For 3800 printers the setup is refreshed with defaults. If spool management functions have been active, all VSE/POWER spool XECBs are deleted. Finally the VSE/AF lock 'ATGATE' is released to allow VSE/AF subtasks to enter the abnormal termination routine too. VSE/POWER terminates itself by issuing the VSE/AF EOJ macro.

Recovery Due To Exit Failure The above described behaviour is different in case the abnormal termination has occurred while being in 'exit-state'. If the active task is found to be in 'exit-state', recovery is tried, meaning VSE/POWER continues to process normally and is not terminated. All changes necessary for this behaviour can be found in module IPW$$AT:

1. Message 1Q2CI may contain type and name of a failing exit instead of a VSE/POWER module and is extended to identify the failing task by task-id and the address of the TCB.
2. The test for being in 'exit-state' is done in the subroutine ATCEXST. Although subroutine ATCEXST indicates 'exit-state', recovery is omitted, if the termination was caused by one of the following reasons (near label AT257AX):
   a. the macro IPW$CNC has been issued
   b. the timer subtask issued a cancel request.
3. The recovery is done in the subroutine ATREXST:
   a. Messages 1Q2KI and 1Q2HI, which are defined locally within IPW$$AT, are issued.
   b. Using the IDUMP macro, a dump is written into the dump sublibary. If the IDUMP macro fails, processing continues without taking a dump. The message 1QC5D, which asks for a device address for printing the dump, is not issued.
   c. The failing exit is marked as 'failed' in the exit-table.
   d. All tasks in the TCB-chain which are in the state 'page-fault-occurred', are set to to the state 'dispatchable', because the page-fault-routine of the supervisor has forgotten any outstanding page-fault after entry into the AB-exit routine.
   e. The lockbyte ATGATE is unlocked to allow processing of further abnormal terminations by IPW$$AT (also used to synchronize the abnormal terminations of the various subtasks, see “Abnormal Termination Processing of VSE/POWER Subtasks”).
   f. Using the SETPFA macro the page-fault-appendage routine is established again, because the page-fault-appendage routine has been de-established:
      1) by the supervisor whenever an abnormal termination of the maintask occurs
      2) by VSE/POWER whenever the IPW$CNC macro has been issued
   g. the page-fault-appendage routine is is de-established by the supervisor whenever an abnormal termination of the maintask occurs
   h. Set indication to stop the running task and return to the address within TCB which was updated by VSE/POWER calling modules (IPW$$LR, IPW$$LW, IPW$$NR2, IPW$$NT) before giving control to the exit routine. At this address a stop-code is set to stop the task using the normal way. Usually the stop-code 'S' is used, but also an additional bit is set to update the stop-messages (1Q33I, 1QX3I, 1QY4I, 1RA8I) with the wording 'DUE TO EXIT FAILURE' at the end of the message or with the appropriate returncode (concerning the messages 1RA9I, 1RB6I). Also at this address, the calling modules unconditionally return to parallel mode for their continuation.

Abnormal Termination Processing of VSE/POWER Subtasks: If the AB-exit is invoked due to abnormal termination of one of the VSE/POWER subtasks, the routine is first locked via 'ATGATE' for serial usage. Then message 1Q2CI with PSW and cancel code is issued, and an IDUMP to the VSE/AF DUMP sublibrary of the VSE/POWER partition is requested calling macro IPW$IDM. If IDUMP-ing fails, the operator will be informed by message 1QC5I. Depending on the type of failing subtask different actions are taken. If it is a asynchronous service subtask, the VSE/POWER task waiting on service by the subtask is posted with the appropriate termination code set in the TCB. If the RJE,SNA subtask abends, the ACB is closed and all RJE,SNA tasks waiting for VTAM posting are reactivated. If the PNET subtask abends, the ACB is closed and the PNET driver is informed about the cancelation of the subtask. Finally the 'ATGATE' lock is released and the subtask is terminated by the DETACH macro. Only in case of the Timer subtask for shared processing, the whole VSE/POWER partition processing is terminated by a CANCEL ALL macro.

In case of abnormal termination of a asynchronous service, the dump or librarian subtask, the subtask is attached again with the next service request.

Termination in an Unattended System: If in an unattended system VSE/OCCF terminated abnormally, and VSE/POWER terminates thereafter, VSE/POWER issues a REIPL macro to bring up the system again. Note this is true not only for the abnormal termination of VSE/POWER but for the normal termination of VSE/POWER, too. In both cases the VSE/POWER routine IPW$$AT (although called abnormal termination) gets control due to the usage of the STXIT macro.

If VSE/POWER terminated abnormally due to an internal cancel issued by the heartbeat task, the REIPL macro is issued at once. Otherwise VSE/POWER updates the REIPL parameters and (if no PEND FORCE issued) waits till all controlled partitions are unbatched. During the cancelation of the controlled partitions, VSE/POWER counted the canceled partitions and compares its counter with the counter which is updated by the end of task routine of the supervisor whenever a partition gets unbatched. The flag IJBPOWT in the system communication region indicates to the supervisor that counting is necessary.

VSE/POWER Multitasking

In order to execute VSE/POWER tasks concurrently, but asynchronously, VSE/POWER incorporates multitasking support. Because this support does not depend on the multitasking (asynchronous processing) support provided in VSE/AF, it is called private multitasking.

Each VSE/POWER task is equipped with a task control block (TCB) created in fixed storage. The TCB is used to establish the identity of the task and to preserve its status when it is not in active control of the central processor.

The task control blocks present at any time in VSE/POWER are linked together by means of next task and previous task pointers to form a logical list called the task selection list. The task selection list is considered to begin and end with the Wait Control Block (WCB), a skeleton TCB located within IPW$$NU (in pfixed SVA) whose function is to delimit the task selection list.

The logical position of each task control block within the task selection list (see Figure 18 on page 59) determines its dispatching priority relative to the other tasks within the list. This priority takes effect only when task selection is entered; once a task is running it will continue to run until it yields control by means of one of the task selection service macro instructions (IPW$WFx) or sustains a page fault. Thus, a higher priority task will not interrupt a running task.

An initial task selection list is constructed by the VSE/POWER initiator (IPW$$I2). This list contains the wait control block, the task control block of the permanent command processor task, and the task control block of the initiator task. This list (or ring) is linked together by forward and backward pointers. All further additions to and deletions from the task selection list are performed by the task management service.

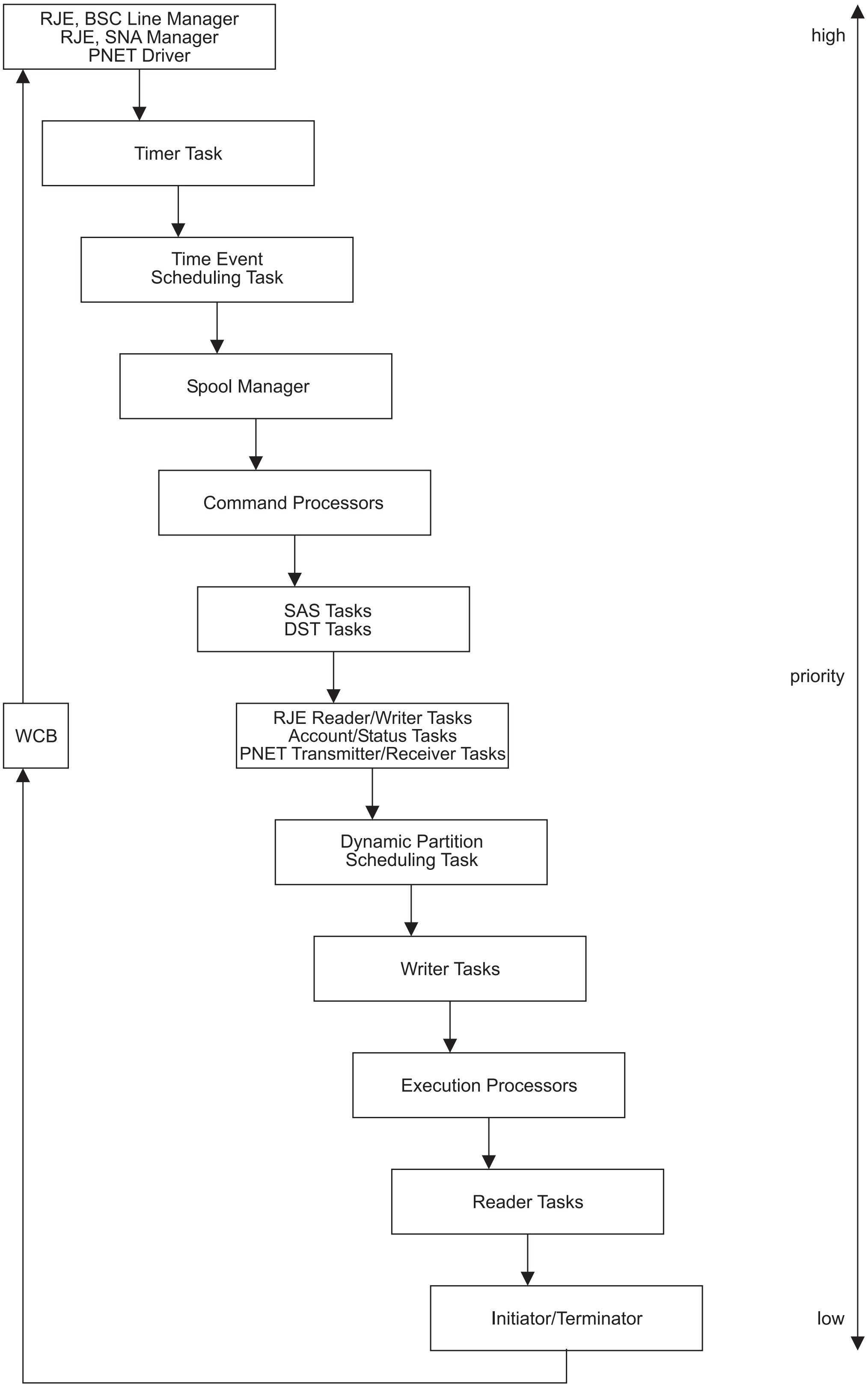


Figure 18. Task Selection List (TSL)

Notes:

1. A writer task started with option VM or SP is chained to the reader tasks and has therefore a lower dispatching priority as the execution processors.
2. A writer task started with option HP is chained to the SAS or DST tasks and has therefore a higher dispatching priority than the execution processors.
3. If SET DYNAL=LOW is specified during VSE/POWER autostart processing the dynamic partition scheduling task is chained to the reader tasks and has therefore a lower dispatching priority than the execution processors of static and dynamic partitions.

VSE/POWER provides three components of task management service:

- Task initiation - attach new task
- Task selection - select next task for dispatch
- Task termination - detach current task.

Each of these components is discussed in the following paragraphs.

Task Initiation

Task initiation is entered from a VSE/POWER task by means of the IPW$ATT (attach new task) macro instruction. The issuing task has already acquired storage for and formatted the task control block which will represent the new task; in particular it has created the task storage descriptor which establishes the task type and identity.

First task initiation pre-determines for all tasks to enter their processing code as a parallel workunit (TCF16NP=OFF), except for tasks of the NP-Must list (PNET SNA Connect/Disconnect and all RJE/SNA Tasks), that have to run always as a non-parallel workunit. For details, refer to “Multiprocessor Support” on page 111. Then task initiation determines the point within the current task selection list at which the new task control block must be inserted, and adjusts the 'previous task' and 'next task' pointers within the task control blocks concerned. The new task is then set into D (dispatchable) state, and return is made to the calling task. This is illustrated in Figure 19.

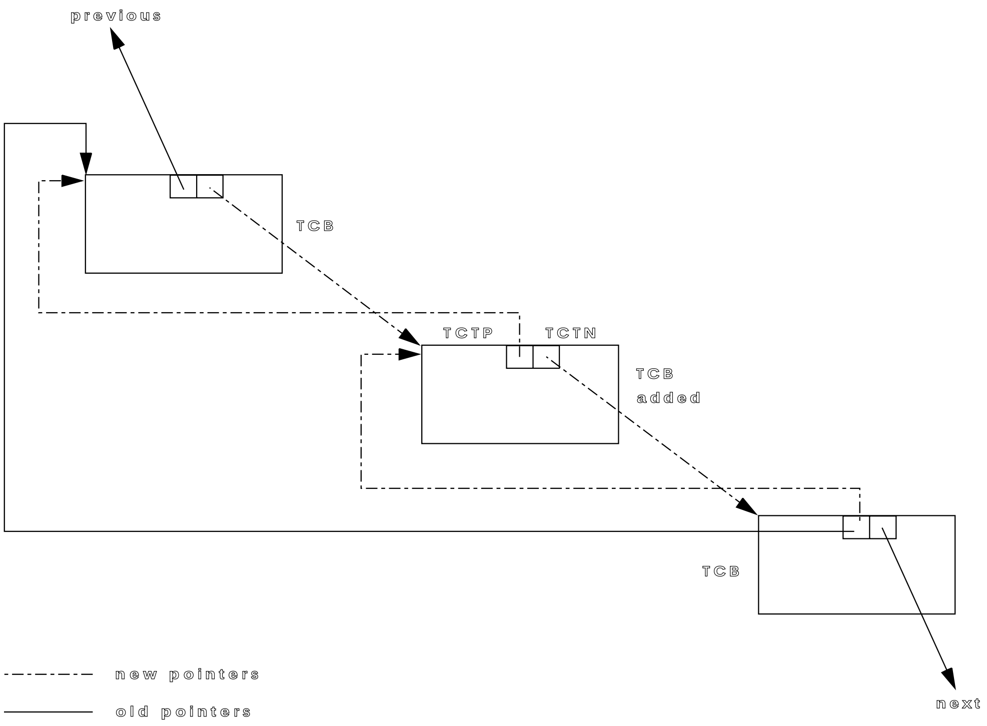


Figure 19. Attaching a Task

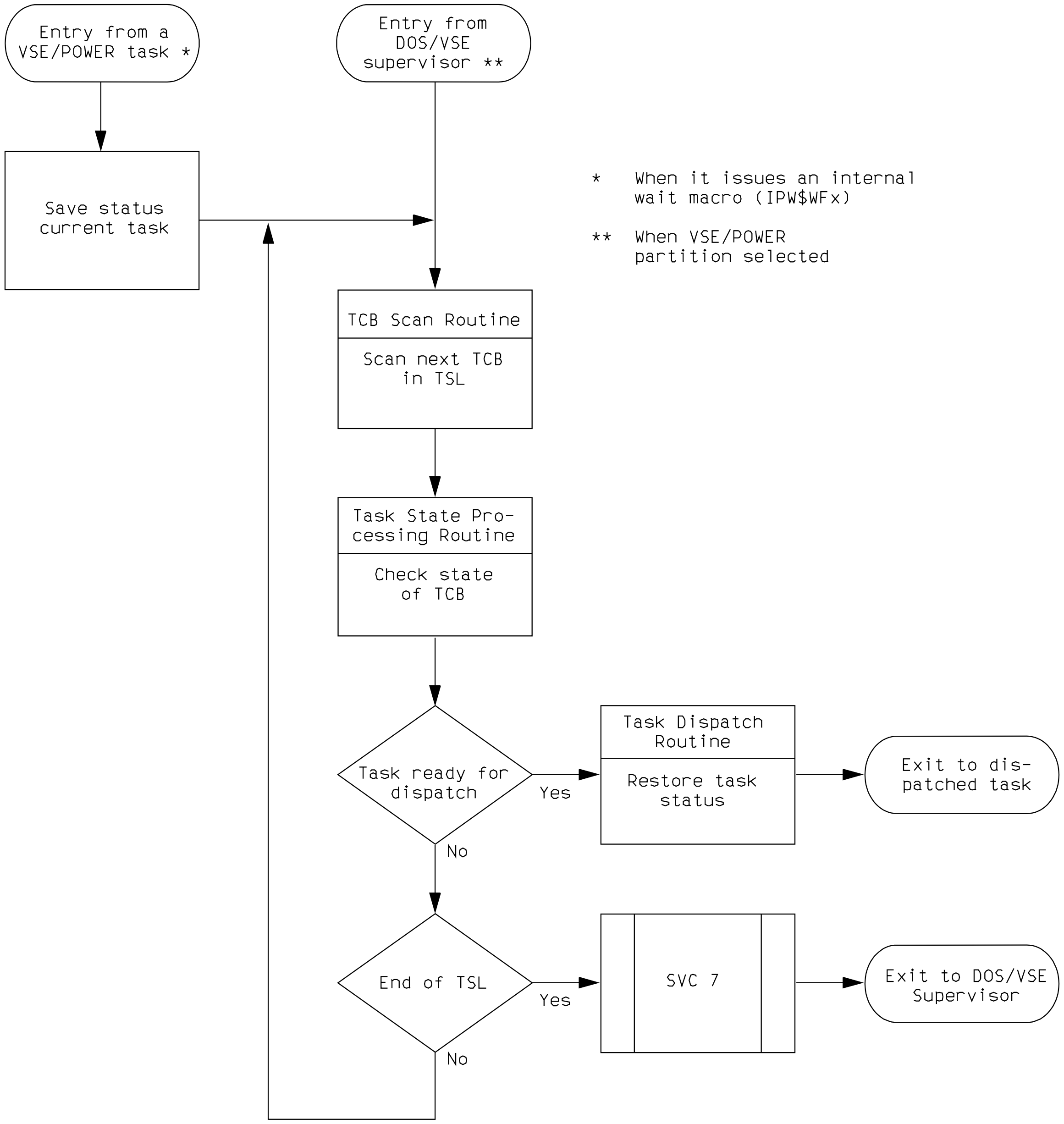

Entry from a VSE/POWER task *
Entry from DOS/VSE supervisor **
Save status current task
* When it issues an internal wait macro (IPW$WFx)
** When VSE/POWER partition selected
TCB Scan Routine
Scan next TCB in TSL
Task State Processing Routine
Check state of TCB
Task ready for dispatch
Yes
No
Task Dispatch Routine
Restore task status
Exit to dispatched task
End of TSL
Yes
No
SVC 7
Exit to DOS/VSE Supervisor

Figure 20. Overview of Task Selection

Task Selection

An overview of task selection is shown in Figure 20. Task selection is entered from a VSE/POWER task when that task yields control to the central processor by means of one of the IPW$WFx (wait for 'X') macro instructions listed below. In each case 'X' represents the task state value to be associated with the task yielding control.

IPW$WFE set E state and wait for external ECB posting.
IPW$WFI set I state and wait for initiation.
IPW$WFO set O state and wait for operator response.
IPW$WFB set B state and wait for posting on RJE,BSC or PNET event.
IPW$WFL set L state and wait for locked resource.
IPW$WFM set M state and wait for multiple control block posting.
IPW$WFQ set Q state and wait for class table posting

IPW$WFX set X state and wait for mixed posting of one ECB and/or class table anchor(s) posting.
IPW$WFC set C state and wait for ECB or CCB posting.
IPW$WFS set S state and wait for ECB posting.
IPW$WFD set D state and wait for re-dispatch.

(The significance of these individual states will emerge in the discussion of the routines that issue the individual macro instructions.)

The status of the task yielding control is saved by storing the current contents of the general purpose registers (and the condition code) in the task register save area of the task control block. This done, the task selection process can begin.

The task selection list is used to address and examine each task control block in turn in order of dispatching priority to determine whether the associated task can be dispatched. This is done by means of the task state value set in the task control block. In addition to the task states listed above, one additional state must be mentioned: P state (page-bound), which is set by the page fault appendage (see "VSE/POWER Appendages") when a task sustains a page fault.

Tasks in the following states are non-dispatchable:

I state - the task is waiting for reactivation.
P state - the task is waiting for a page-in operation.
O state - the task is waiting for operator response.

Tasks in the following states are conditionally dispatchable. A further test or tests must be performed to determine whether the condition has been satisfied and the task is in fact ready for dispatch.

L state - the task is waiting for a locked resource.
E state - the task is waiting for external ECB posting.
S state - the task is waiting for ECB posting.
C state - the task is waiting for ECB or CCB posting.
Q state - the task is waiting for class table posting or multiple XECB posting.
M state - the task is waiting for any of a set of ECB or CCB postings.
B state - the task is waiting for a RJE,BSC or PNET event.
X state - the task is waiting for posting of one ECB and/or class table anchor(s)

Tasks in the following state are unconditionally dispatchable:

D state - the task is ready for immediate dispatch.

As soon as a dispatchable task is found within the task selection list, the active VSE/POWER Maintask, which is about to give life to the dispatchable Private Subtask ...

1. requests NP/PA Mode switch according to the TCF16-workunit, which has been pre-determined for a task been just attached, or has been recorded for a task been interrupted before
2. the general purpose registers (and condition code) are restored from the task register save area of the task control block
3. the task is set into R state (running)
4. and execution of the task is resumed from the point at which it previously ceased with either Amode-24 or even Amode-31 (as been recorded in the TCB at a previous page fault).

If running in /370 mode and the task to be dispatched is either an execution or spool manager task and the partition serviced by the task resides in a different address space, VSE/POWER switches to the appropriate address space before giving control to the task.

If the entire task selection list is scanned without any task being found to be dispatchable, the task selection service issues an SVC 7 to pass control to the VSE/AF supervisor. Additionally the **no-work-to-do** ECB is posted when no task is waiting for a locked DMB or, if the account file is shared, for a locked ACB. VSE/POWER will wait till the occurrence of some related event (I/O completion, for example) causes VSE/AF to return control to the task selection service. The entire task selection process is then repeated.

Task Termination

Task termination (Figure 21) is entered from a VSE/POWER task by means of the IPW$DET (detach current task) macro instruction.

Task termination removes the task control block of the current task from the task selection list by adjusting the 'previous task' and 'next task' pointers within the neighboring task control blocks within the list. The storage occupied by the eliminated task control block is returned to the system, and control is then passed to task selection to determine the task next to be dispatched.

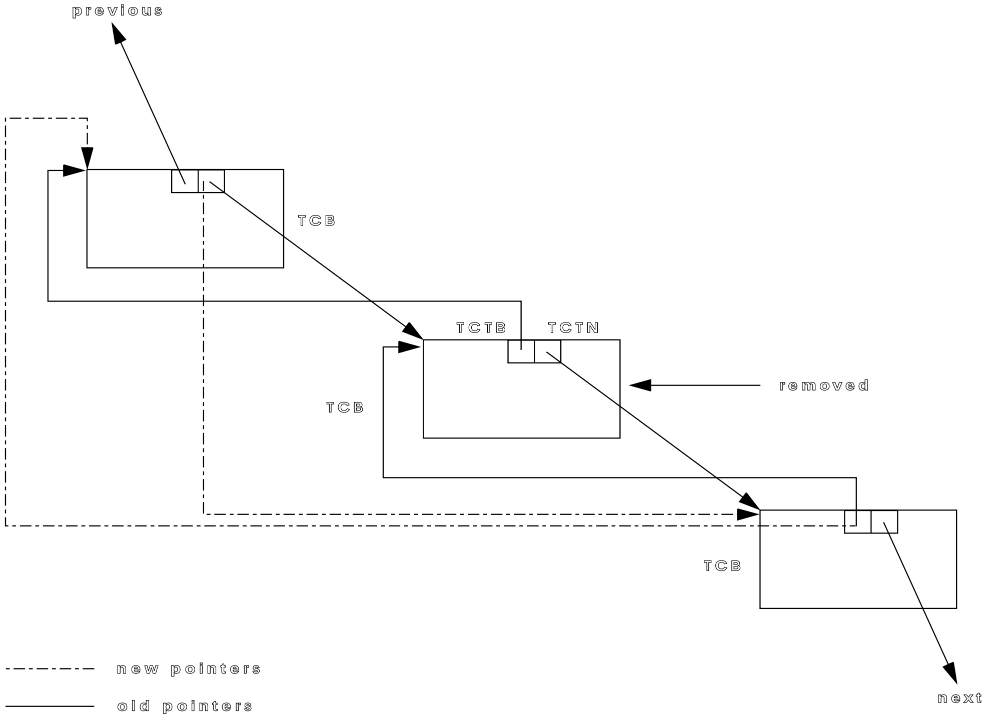


Figure 21. Detaching a Task

Reader, Execution Processor, and Writer Tasks

The data flow throughout the reader, execution processor, and writer task is summarized by Figure 22.

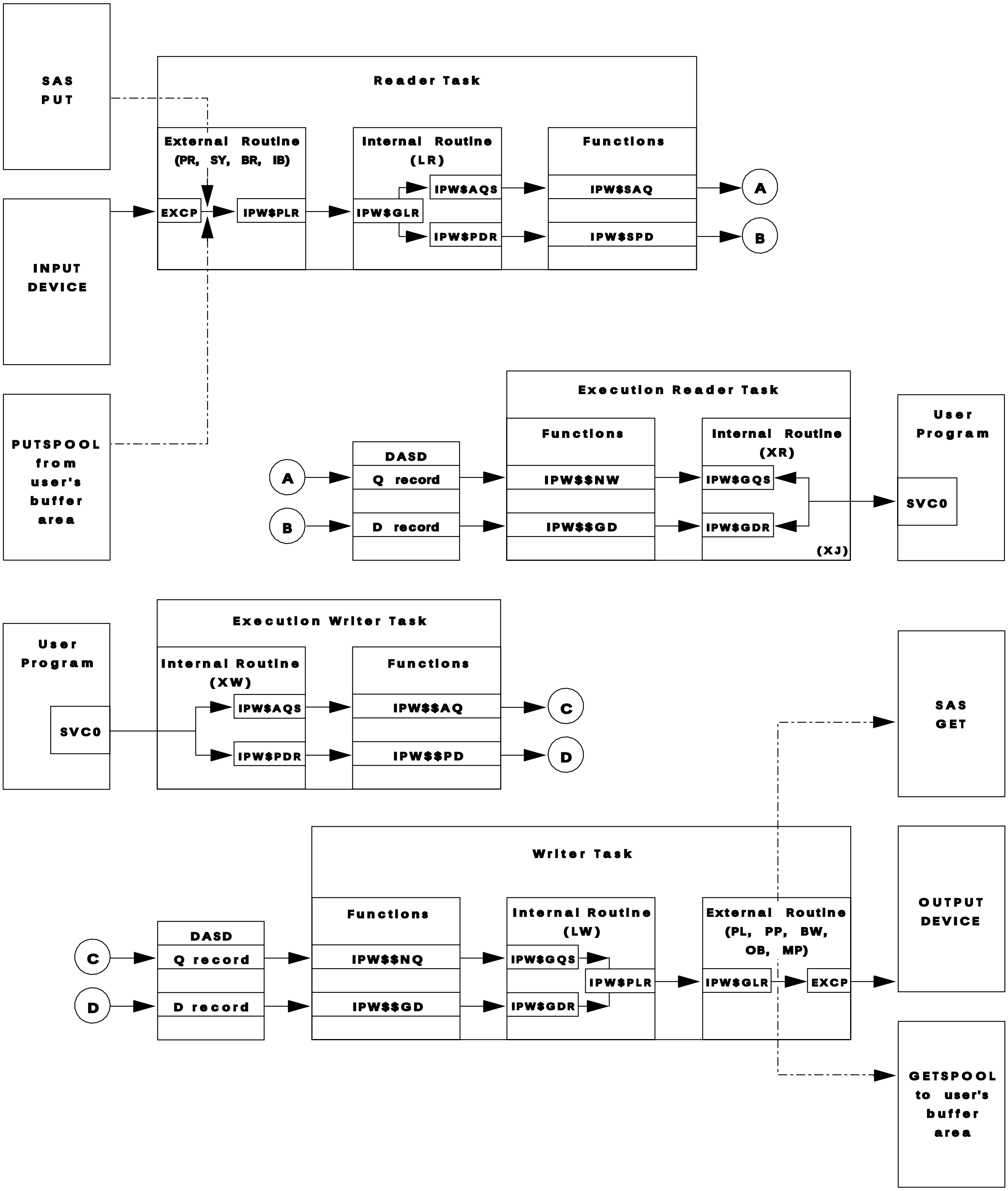


Figure 22. Data Flow Throughout the Spooling Process

Reader Tasks

The reader task is executed by a physical reader routine (PR) and logical reader routine (LR). These routines pass control to each other through a logical record interface. At unit exception, the task places itself in a dormant state, releasing as much work space as possible. "Hot reader" support enables a dormant task to continue without a PSTART command, if new input has become available (refer to “Hot Reader Appendage”).

Physical Reader (IPW$$PR): The IPW$$PR routine is entered when a reader task is invoked by a PSTART command, or when an unsolicited device-end interrupt occurs while the task is in a dormant state (hot reader support). Special work areas will be allocated at entry time and initialized according to the supported physical device (see Figure 23). The work areas can be released by the termination routine IPW$$TR.

The IPW$$PR routine performs the physical input for one or more devices and establishes the linkage with the IPW$$LR routine so that, on request, each logical record can be passed over the interface to the IPW$$LR routine. Each input operation will handle a number of records by means of command chained CCWs (refer to "Physical Data Record Area" in Chapter 5). The input operation is performed with real addresses in the CCWs (/370 mode only).

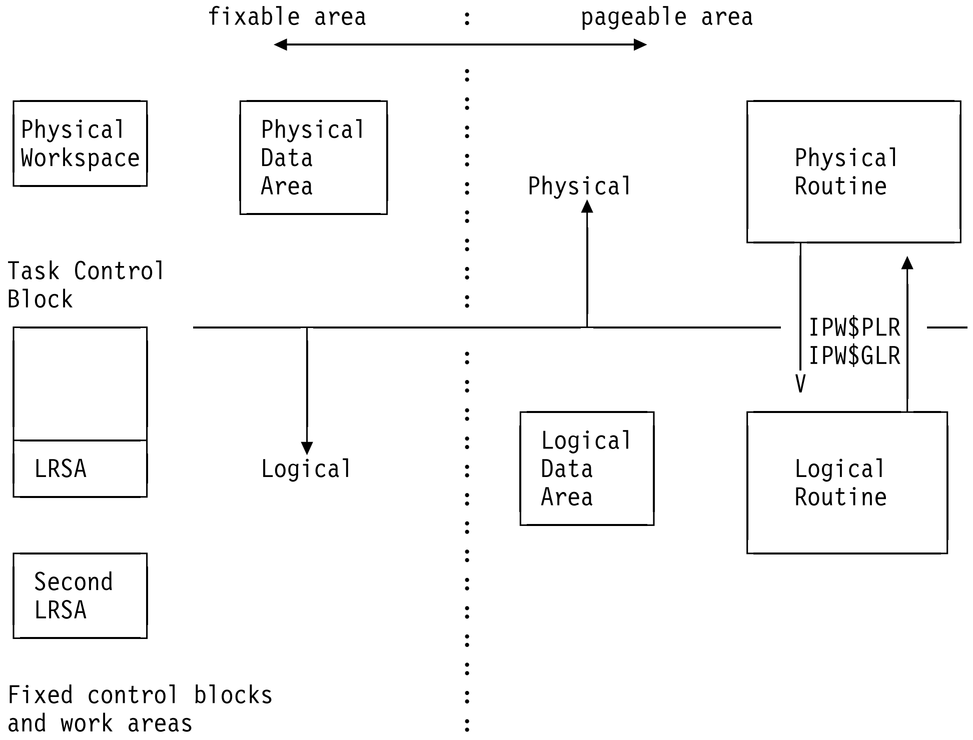
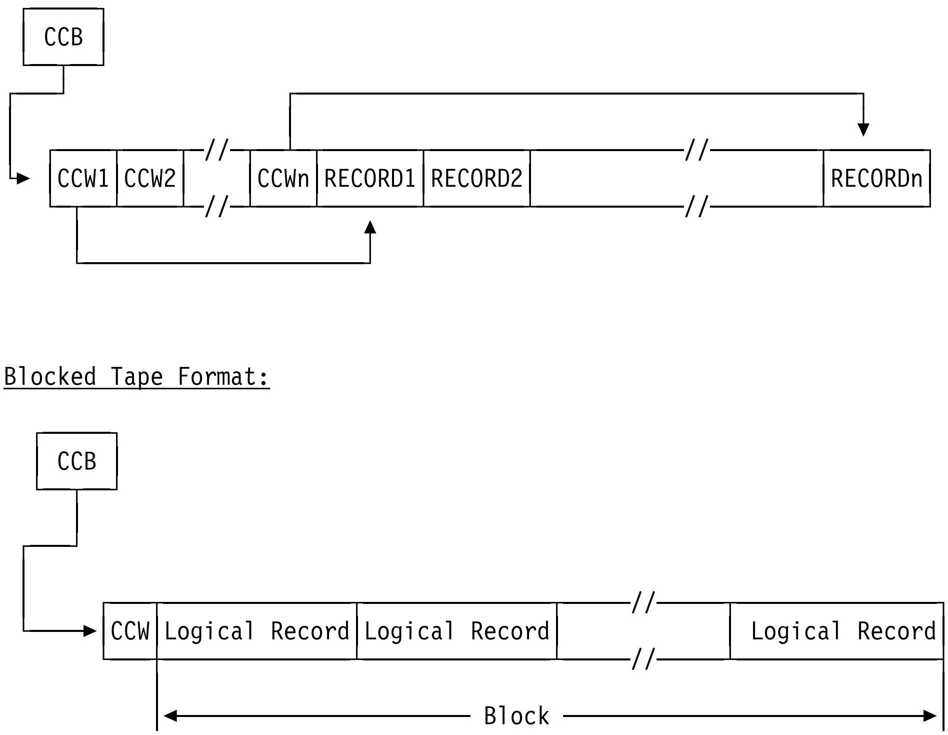


Figure 23. Physical and Logical Work Areas

SYSIN Tape Reader (IPW$$SY): The IPW$$SY routine is entered when a tape reader task is invoked by a PSTART command. An area of 4K bytes in the partition GETVIS area will be used for tape input (see Figure 24). When started, the tape reader task requests GETVIS space and will wait until the space is available. The number of SYSIN tape readers is only limited by the number of physical tape units, or the amount of GETVIS space available. Record format CCW chaining may be performed, which means unblocked format causes a number of CCWs chained to read a number of records with one input operation.

The tape reader task establishes linkage to the logical reader (IPW$$LR) and passes each logical record in turn to the logical reader using the interface (IPW$PLR).


Unblocked Tape Format:

Blocked Tape Format:

Figure 24. Physical Data Area - GETVIS Space

Physical 3540 Diskette Reader (IPW$$ER): This routine is entered via the logical reader when a RDR statement is encountered in the input stream, or via task selection as a result of a PSTART command issued for the diskette reader only. It reads data from the physical diskette reader associated with the reader task.
If the routine is entered from the logical reader and no diskette unit is assigned (dynamic RDR support), the PUB table is scanned for a free, operational 3540 device. If no such unit is available, the job is flushed and the operator is informed via message 1Q90I. Otherwise the diskette unit remains assigned to the job until end of job is encountered.

Logical Reader (IPW$$LR): The first time the routine is entered, it reserves work space for the queue record area and acquires a queue record from the free queue record chain (via IPW$RQS macro instruction).

The values may be overwritten by specifications in the JECL statements (* $$ JOB and * $$ CTL). A job header record is set up and passed to the put data record routine. Records passed via the logical record interface will be passed in turn to the put data function routine (IPW$$PD) for writing to the data file. The general purpose byte in the record control word (RCW) of the TCB indicates what action is to be taken by the IPW$$PD routine.

General-purpose byte posted by the logical reader:

End of data	for last record for this job entry
End of block	in case of unexpected end of input (expected delimiter not encountered, or last record of block).

The routine provides a user exit. It enables a user-written routine to examine each JCL and JECL statement, including any continuation statements, and delete or insert records in the job stream. Before entry to the user exit a default switch to non-parallel (NP) mode is done to allow for Supervisor Control Block update by the exit. If however the loading conditions of the exit specify 'PA', this extra switch is suppressed. Upon return from the user exit an unconditional switch to parallel mode is done. For details refer to “Multiprocessor Support” on page 111.

If the last record for the currently processed job entry is passed, a skeleton job trailer record is passed to the put data record routine and the add queue entry function is invoked (via the IPW$AQS macro) to add the queue entry to the appropriate class chain according to its priority.

Note: When the logical reader encounters the first record with a record length other than 80 or with a change of characteristics, a data set header record, which describes the characteristics of the following data, is built and passed to the put data record routine.

Execution Processor Tasks

The execution processor tasks are:

- Execution reader task (IPW$$XRE and IPW$$XJ)
- Execution writer tasks (IPW$$XWE)

The dynamic partition scheduling task acts as a static 'hyper' execution task and selects jobs from the RDR queue for processing in a dynamic partition. For more details on dynamic partition support see later on in this chapter.

Each serviced VSE/AF partition has a partition control block. For the 11 static partitions the partition control blocks are reserved behind the VSE/POWER nucleus during VSE/POWER initialization. Each control block has space for the maximum number (29) spool devices (device entry list).

The partition control block contains also header information pertinent to the partition itself. Each device entry relates to a single real or dummy physical device specified in the PSTART command for a static partition given by the operator.

The first device entry within each partition control block describes the reader device for that partition. If the partition is a writer-only partition the device described by the reader entry is the system console device. Further device entries describe the list devices and punch devices for the partition.

Each device entry is used to pass information from the user partition to the VSE/POWER execution processor task which is responsible for the emulation of that device.

An execution reader task is started for each partition at the time at which the partition is brought under VSE/POWER control. It continues to run until the partition is returned to VSE/AF control by means of a PSTOP command.

This task is responsible for servicing all read requests addressed by the user program to the partition read device designated at partition PSTART time. It is additionally responsible for recognizing the first request addressed by each job executed within the partition to each of the partition list and punch devices designated at partition PSTART time, and initiating an execution writer task to service the further program requests addressed to that device.

Until end of job the execution tasks proceed concurrently but asynchronously. When the execution reader task detects an end-of-job condition it posts a stop condition to each of the subordinate tasks that it started. It then waits until each of these tasks detaches itself in turn.

If no other queue entry can be processed the execution reader task will place itself in a wait state, after a message is issued. When a PSTOP command is issued, the execution reader task and its subordinate tasks will eventually be detached after processing the current queue entry.

Execution Reader Routine (IPW$$XRE) This routine will emulate the user channel program input requests for the reader device. To service these requests a data record is kept available throughout the process of this routine. Records are retrieved via IPW$GQS and IPW$GDR macro instructions. The routine does the following:

- Holds a copy of the job header record in storage anchored to the partition control block of the partition concerned.
- Informs the supervisor that VSE/POWER intercepts read requests for 3540 diskette(s), when indicated in the job header record.
- Intercepts first request for output of the user channel program. Acquires storage for the queue record area and data set header record, initializes them with the VSE/POWER defaults and the information obtained from the job header record. Both areas are then anchored to the TCB of the new execution writer task and the task is then attached.
- Handles all input requests from the user channel program.
- In case of a writer-only partition, analyzes JECL statements from a console read/write operation and starts a writer task.
- Indicates termination of a writer task once a queue entry has been processed.
- When an SVC 90 is encountered, real storage is reserved in the length of the total execution account record plus the length of the user data; the user data are then moved into the account record.
- When an SVC 91 is encountered, storage for an execution account record is acquired, if not already done so by a previous SVC 90. The account record is then initialized with values extracted from the VSE/AF accounting tables and the queue record; finally the account record is written by means of the IPW$PAR macro instruction.
- When an SLI JECL statement is encountered, a parameter list is built and passed to the SLI processing routine; this routine then calls the librarian subtask in order to locate the member in the source statement library.
- When PUN, LST, or PRT JECL statements are recognized, terminates the appropriate writer task, builds a new queue record and data set header record, and starts the writer task again.
- Completes the job trailer record with accounting information at end of job time.

See Figure 25 on page 71 for overview of different segmentation execution flows.

Execution JECL Scan Routine (IPW$$XJ): This routine parses the JECL statements, checks the validity of the parameters specified and updates the various VSE/POWER control blocks accordingly.

The following statements are checked - JOB, LST, PUN, SLI and DATA. The JECL RDR, CTL and EOJ statements are ignored.

See Figure 25 on page 71 for overview of different segmentation execution flows.

Execution Writer Routine (IPW$$XWE): At entry of the execution writer routine, the execution reader task has already reserved queue space and initialized the queue record area and data set header record either with VSE/POWER defaults or with information obtained from the
* $$ LST|PUN statement. The data set header record is anchored to the execution writer task TCB.

Space is reserved for the data buffer for the output records. If tape spooling was requested, a tape control block is set up by executing the IPW$OTP macro instruction. The tape control block is anchored to the TCB of the task. If the device being spooled is a 3800, the data set header record is completed with the defaults setup by means of the SETDF command for the appropriate 3800 device (the defaults are extracted from the PUB2 area).

For performance reasons, a stack of up to 30 internal FCB image representations is maintained in virtual storage. The stack can be deleted by the PDELETE FCB command in order to force stack recreation. If an FCB name is specified in the data set header record, this stack is scanned for a matching FCB name. If a match is found, the internal representation is copied into the TCB of the execution writer task and no FCB image load from the library is performed.

If no match was found in the internal stack, then the FCB image is loaded from the library and converted to the VSE/POWER internal representation which is then added to the stack in a FIFO manner.

The job header record and data set header record are then passed to the put data record routine.

If a request from the user program is found in the task list entry of the partition control block, the user channel program is emulated. If no entry is found the task enters a wait state for further user program requests or for a segmentation command (IPW$WFM).

Each CCW is checked for validity and user data is transferred to the data file by invoking the put data record function.

At termination of the task, which is controlled by the execution reader task (stop code), the job trailer record, set up by the execution reader task and anchored to the partition control block, is passed to the put data routine and the current queue entry is added to the appropriate class chain by invoking the add queue record function. The data buffer and all virtual storage acquired by the task are released and the task detaches itself.

Output segmentation is driven by command (PSEGMENT or PALTER ...,SEGMENT=...) or by count (as specified in JECL) or by the user program (via an FCB buffer load, or by issuing a SETPRT or IPWSEGM or SEGMENT macro) and is established through formation of a new queue entry. The former queue entry is added to the appropriate class chain.

If checkpointing is requested (indicated via the * $$ LST statement), the record number associated with a page for LST or card for PUN output is recorded in the queue record whenever the specified checkpoint interval is reached.

See Figure 25 on page 71 for overview of different segmentation execution flows.

Segmentation Considerations: See Figure 25 on page 71 for overview of different segmentation execution flows.

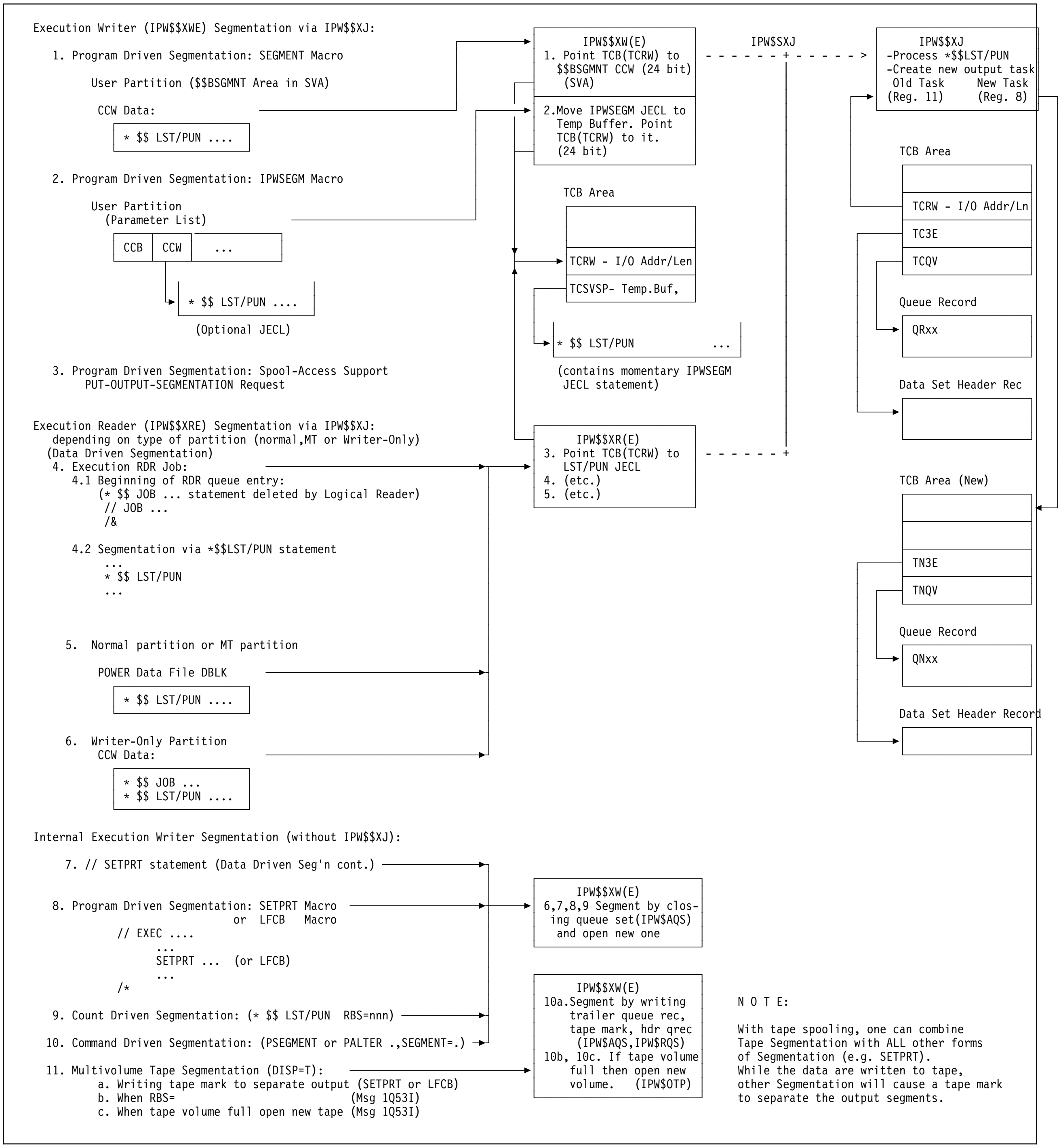

Execution Writer (IPW$$XWE) Segmentation via IPW$$XJ:
1. Program Driven Segmentation: SEGMENT Macro
User Partition ($$BSGMNT Area in SVA)
CCW Data:
* $$ LST/PUN ....
2. Program Driven Segmentation: IPWSEGM Macro
User Partition (Parameter List)
CCB
CCW
...
* $$ LST/PUN ....
(Optional JECL)
3. Program Driven Segmentation: Spool-Access Support PUT-OUTPUT-SEGMENTATION Request
Execution Reader (IPW$$XRE) Segmentation via IPW$$XJ: depending on type of partition (normal,MT or Writer-Only) (Data Driven Segmentation)
4. Execution RDR Job:
4.1 Beginning of RDR queue entry: (* $$ JOB ... statement deleted by Logical Reader) // JOB ... /&
4.2 Segmentation via *$$LST/PUN statement ... * $$ LST/PUN ...
5. Normal partition or MT partition
POWER Data File DBLK
* $$ LST/PUN ....
6. Writer-Only Partition CCW Data:
* $$ JOB ...
* $$ LST/PUN ....
Internal Execution Writer Segmentation (without IPW$$XJ):
7. // SETPRT statement (Data Driven Seg'n cont.)
8. Program Driven Segmentation: SETPRT Macro or LFCB Macro
// EXEC .... SETPRT ... (or LFCB) ... /*
9. Count Driven Segmentation: (* $$ LST/PUN RBS=nnn)
10. Command Driven Segmentation: (PSEGMENT or PALTER .,SEGMENT=.)
11. Multivolume Tape Segmentation (DISP=T):
a. Writing tape mark to separate output (SETPRT or LFCB)
b. When RBS= (Msg 1Q53I)
c. When tape volume full open new tape (Msg 1Q53I)
IPW$$XW(E)
1. Point TCB(TCRW) to $$BSGMNT CCW (24 bit) (SVA)
2.Move IPWSEGM JECL to Temp Buffer. Point TCB(TCRW) to it. (24 bit)
TCB Area
TCRW - I/O Addr/Len
TCSVSP- Temp.Buf,
* $$ LST/PUN ...
(contains momentary IPWSEGM JECL statement)
IPW$$XR(E)
3. Point TCB(TCRW) to LST/PUN JECL
4. (etc.)
5. (etc.)
IPW$SXJ
IPW$$XJ
-Process *$$LST/PUN
-Create new output task
Old Task (Reg. 11)
New Task (Reg. 8)
TCB Area
TCRW - I/O Addr/Ln
TC3E
TCQV
Queue Record
QRxx
Data Set Header Rec
TCB Area (New)
TN3E
TNQV
Queue Record
QNxx
Data Set Header Record
IPW$$XW(E)
6,7,8,9 Segment by closing queue set(IPW$AQS) and open new one
IPW$$XW(E)
10a.Segment by writing trailer queue rec, tape mark, hdr qrec (IPWAQS,IPWRQS)
10b, 10c. If tape volume full then open new volume. (IPW$OTP)
N O T E:
With tape spooling, one can combine Tape Segmentation with ALL other forms of Segmentation (e.g. SETPRT). While the data are written to tape, other Segmentation will cause a tape mark to separate the output segments.

Figure 25. Execution Segmentation Data Flow Overview

3800 Printer Considerations: Any request to alter the printer setup, either via a // SETPRT statement or a SETPRT macro instruction, is routed to SETPRT. When SETPRT determines that the device is being trapped by VSE/POWER, it passes the SETPRT parameter list to VSE/POWER after a basic validation. This is done by issuing an I/O to the device with an 'FD' channel command operation code, and with the data area address pointing to the SETPRT parameter list. The execution processor recognizes the 'FD' operation code as a valid command for the 3800. SETPRT handling is illustrated in Figure 26.

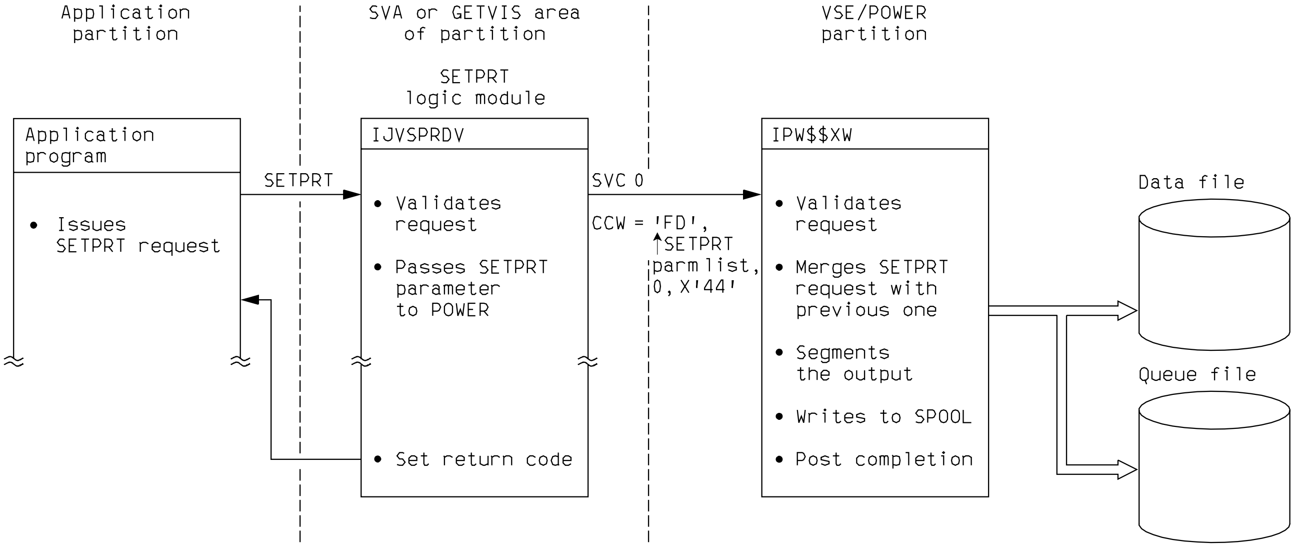


Figure 26. SETPRT Handling

The execution processor maintains a control block, called data set header record, which contains the current printer setup of the device being spooled. When a SETPRT parameter list is encountered by the execution processor, the printer setup is updated, which means the new setup request is merged with the previous one.

When the BURST, FORMS, FCB, FLASH, or copy group specifications have been changed, the output is segmented (that is, the output entry is closed and added to the class chain according to the priority; then a new output entry is created with the same jobname and job attributes but with a different job number, in order to facilitate queue manipulation by the operator. The job header record and data set header record are then written as first records in the new list queue entry.

Whenever the execution processor detects that a valid CINDX value (other than 0 or 1) was specified in the SETPRT parameter list, it assumes that the user will manage the copy group handling by himself.

The execution processor creates a new output LIST entry with the same job attributes and sets the transmission count to one.

When a SETPRT parameter list contains an FCB specification, the FCB image is loaded from the library and the internal representation of the page format is updated. The data set header record is updated accordingly. The FCB image is validated for accuracy. If a 3800 FCB image is invalid, a message (1Q54I) is written to the operator and the LTAB or default LTAB (reduced in length by 6 lines equal to 1 inch) is used. The LTAB specification is assumed as the internal representation of the FCB.

The following 3800 CCW operation codes are not accepted by VSE/POWER (execution writer) and cause the channel program check and the unrecoverable I/O error flags in the CCB to be posted:

- Load translate table (X'83')
- Load character module WCGM (X'53')
- Load forms overlay sequence control (X'43')
- Load copy number (X'23')
- Load graphic character modification (X'25')
- Load copy modification (X'35').

If the user is prepared to accept unrecoverable errors, control is returned to him in the normal way; otherwise, the user partition is canceled.

The "clear printer" (X'87') 3800 CCW operation is ignored.

Writer Tasks (List and Punch)

The writer task is executed by a physical routine (IPW$$PL or IPW$$PP) and a logical routine (IPW$$LW). These routines pass control to each other through a logical record interface. If no next job is available, the task places itself in a dormant state, releasing as much work space as possible.

Physical List and Punch (IPW$$PL and IPW$$PP): These routines are entered when a list or punch task is invoked by a PSTART command. At entry, special work areas are allocated and initialized according to the supported physical device (see Figure 23 on page 66). The physical list routine sets up the printer when appropriate with the requested forms control buffer (FCB) and universal character set buffer (UCSB). The requested FCB name, obtained from the Data Set Header record, or if none was specified, the default FCB name, is compared with the one currently loaded in the printer. (This name is saved in the printer extension area anchored to the TCB of the writer task.) If the names do not match, asynchronous service is invoked by means of the IPW$IAS TYPE=SERVICE macro to perform the FCB load. This is required because the load may involve waits which are not allowed by the VSE/POWER main task. If the FCB load fails, message 1Q54I is issued and a branch is made to the task terminator routine (IPW$$TR).

The UCSB name obtained from the data set header record is compared with the UCSB already on the printer and the options (block/unblock data checks, fold/unfold) are also checked for any change. If a change is found then the UCSB is loaded and message 1QA7A is issued to request the operator to mount the proper print train. Processing is halted until re-started or terminated by the operator. If an error occurred while loading the UCSB image, message 1Q54I is issued and a branch is made to the task terminator routine (IPW$$TR). The work areas are released by the termination routine IPW$$TR.

Both routines perform physical output. On request, the linkage allows logical records to be received in turn over the interface from the logical writer routine.

SETPRT processing is illustrated in Figure 27. If a SETPRT parameter list was passed by the logical writer, the partly filled print buffer is emptied and a IPW$IAS TYPE=SERVICE macro instruction is issued to perform the printer setup. Additionally if the DEBUG option was specified in the SETPRT parameter list, SYSLST is temporarily assigned for the duration of the setup. Each output operation will print or punch a number of records by means of command-chained CCWs (see "Physical Work Space"). The output operation is performed with real addresses in the CCWs (EXCP real, System/370 mode only).

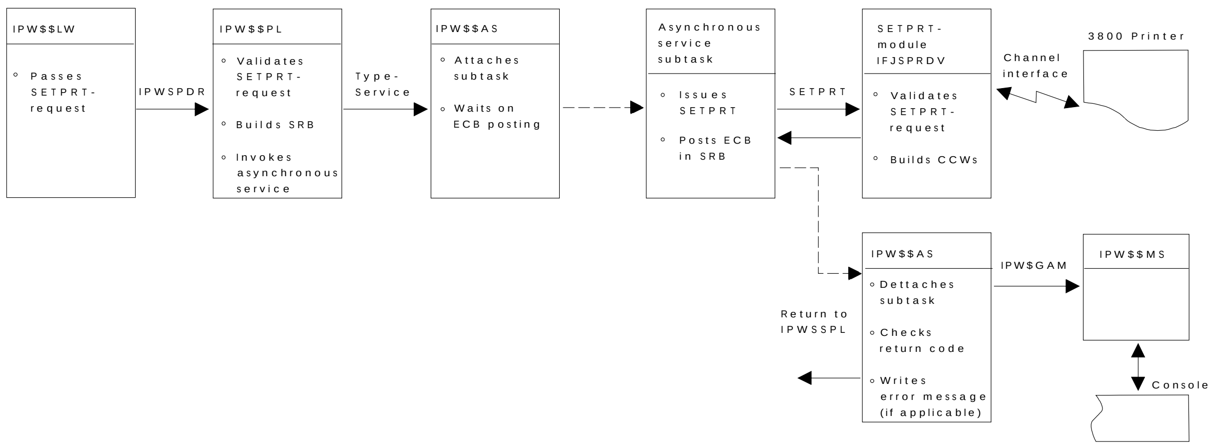


Figure 27. SETPRT Request Processing Flow

Logical Writer (IPW$$LW): A new queue entry is addressed by invoking the get next queue entry function. If no queue entry is eligible, a physical writer task is placed in a wait state until a new eligible queue entry is added or an existing queue entry becomes available. In all other cases the logical writer routine returns to its caller with an indication that there is nothing to do.

The routine ensures that the controlled printer/punch or even remote station is set up with the requested forms. The forms id obtained from the queue record is compared with the one in the TCB of the writer task that specifies the actual setup. If a mismatch requires operator intervention (for 3800, a mismatch of forms, flash or burst status), message 1Q40A or 1QA5A is issued. For a 4248, the mount forms message is also displayed at the printer's display panel. The operator must then either perform the setup and use the PGO command to continue processing, or else stop the writer task or flush the output via the PSTOP/PFLUSH command respectively.

A warning message (1Q41I) is issued if a different printer/punch device is used at physical print/punch time as was used at execution time. The operator can then decide whether to continue with the output or to flush the output. If he decides to continue, all illegal commands passed to the physical device are ignored and the output may contain invalid data or may lose records.

If the physical task is started as a 'VM writer' task, the following VM-CP spool command is issued at the beginning of the queue entry processing to supply VM-CP with more descriptive information about the output queue entry.

```
SPOOL cuu TO {user id|SYSTEM} CLASS c COPY nnn FORMS ffff
```

If the target user id is unknown in the VM-CP directory, message 1QAAI is issued, and the queue entry is placed back in the appropriate VSE/POWER queue in 'hold' disposition.
Only one copy of the queue entry is passed to VM-CP, even so multiple copies are requested. After the queue entry is completely processed, a VM-CP close command is issued to close the queue entry whereby the VSE/POWER job name and number are passed as VM file name and file type respectively. The DIST value is passed also to VM in the CLOSE command, if a DIST value was specified in the * $$ LST statement.

Start separator pages/cards are produced, if applicable at the beginning of each output and between copies.

A logical record is retrieved from the data file by invoking the get data record (IPW$$GD) function.

Job header and job trailer records are ignored by the logical writer routine. If the record just obtained was a data set header record and the device being used is a 3800 printer, the SETPRT parameter list is extracted from the VSE/POWER section of the data set header record. If no VSE/POWER section is present, a SETPRT parameter list is built from the information of the 3800 section of the data set header record, if present, else a default SETPRT parameter list is built.

Note: The SETPRT parameter list is passed as a 'FD' record to the physical routine.

The logical record is passed over the interface to the physical routine. Data records with an ASA carriage control character are divided into two records to perform the requested print/punch function.

- The ASA carriage control character is converted into the corresponding machine op-code and passed as a control record to the physical routine.
- The data record is passed without the ASA character to the physical routine. For punch data, each record is converted into a write, feed select stacker 1 (X'21') operation.

The general purpose byte is tested for following action to be taken:

- Normal record: Retrieve the following logical record.
- End of data record: Delete the queue entry by invoking the delete queue entry function. Additionally for a 3800 printer an end-of-output CCW is issued to effect offset stacking. A clear printer CCW is issued, if requested, to ensure that all data in the printer buffer are printed before the queue entry is purged from the queue.

  End separator pages/cards are produced if requested. If more copies are requested, the routine is repeated until all copies have been produced. The next queue entry function is then invoked to address the next queue entry, if any is available.

If restart of an output queue entry is requested by the operator by means of the PRESTART command or via the SAS interface, the logical writer scans the DBLK groups, locating the DBLK group, which contains the record, page or line where to restart. This is done by using the following fixpoints

1. Start of the queue entry as determined from queue record
2. Current position in certain DBLK group, when restart requested
3. End of the queue entry as determined from queue record

and by calculating the shortest way from start/current/end position to the restart target. On this way either the DBLK group forward chain based on the Spool Environment Record (SER) or the DBLK group backward chain based on the Spool Environment Header (SEH) is used. Forward reading, for example, is done by reading the last DBLK of the DBLK group and examining the SER if the record to start with is within the DBLK group. If not, the last DBLK of the next DBLK group, pointed to by the SER, is read in. This continues until the DBLK-group is found. Backward reading follows the SEH records and works correspondingly, keeping in mind that the SER of a previous DBLK group contains the same spooling values (line, pages, setup) as the SEH of the next DBLK group. If the target is found and the printer is a 3800, the last 'printer setup', contained in the previous SER, is re-established. For a 4248 horizontal copy is established according the information located in the previous SER. Then the record in question is located, the requested printer set up is performed and processing continues.

The routine provides a user exit. It enables a user-written routine to examine each record before printed or punched, passed to a RJE workstation or passed to a DDS. The routine may change, delete records or insert new ones. Before entry to the user exit a default switch to non-parallel (NP) mode is done to allow for Supervisor Control Block update by the exit. If however the loading conditions of the exit specify 'PA', this extra switch is suppressed. Upon return from the user exit an unconditional switch to parallel mode is done. For details refer to “Multiprocessor Support” on page 111.

Offload Reader/Writer Tasks (IPW$$OF)

The offload task is entered when a reader or writer task is invoked by a POFFLOAD command. The offload task performs one of the following functions:

- SAVE function
- LOAD function
- BACKUP function
- PICKUP function
- SELECT function
- BACKUPnn/SAVEnn/PICKUPnn function

SAVE Function: This function retrieves one or more spool entries in dispatchable state from the VSE/POWER queues (RDR, LST, PUN or XMT) according to the specified class(es) and puts them on tape.

If the SAVE ALL function has been used or a * has been specified as the class, then all classes are searched for eligible entries. Eligibility in this case means that the SYSID, REMID, and DESTID are ignored when considering whether an entry is eligible.

The queue records and data blocks retrieved remain unchanged (apart from SEH or SER records removed) on tape and can later be restored to the VSE/POWER spooling files using the 'load' function.

Queue entries are separated from one another by a single tape mark so that after the function has been performed the tape looks like an unlabeled multifile volume. Only queue entries from one specified queue will be processed at a time, unless the SAVE ALL function was specified.

LOAD Function: This function will be used to restore queue entries, residing on tape, to the VSE/POWER queues. Only those entries can be restored which match with the specified queue identifier of the POFFLOAD command. If the LOAD ALL function is requested then all queue entries will be restored from the tape. The restore function operates independently of block size, which means that different DBLK sizes may be used between save and load time. The queue entries will be restored according to their class and disposition, unless the operator specifies a class.

The POFFLOAD tape format which is identical to the spool tape format is shown in Figure 28.

BACKUP Function: This function writes each queue entry of individual classes or of a given queue on tape, regardless of what the disposition of the queue entry is. Queue entries are separated from one another by a single tape mark. The function can be used to archive the VSE/POWER spool files. The queue entries are not deleted from the queues after they have been written onto tape. Only one POFFLOAD BACKUP task can run at any time since the DMB is exclusively locked for the duration of the backup processing.

PICKUP Function: Similar to BACKUP, this function writes each queue entry of individual classes or of a given queue on tape, regardless of what the disposition of the queue entry is. The function begins by scanning the queue file for eligible spool entries, and setting a PICKUP flag indicating that the spool entry will later be written to tape (if still available). The function can be used to archive the VSE/POWER spool files. The queue entries are not deleted from the queues after they have been written onto tape. Only one POFFLOAD PICKUP task can run at any time, but unlike BACKUP, the DMB is not exclusively locked for the duration of the processing. Similar to SAVE, the function may run in parallel with other tasks which require the DMB.

SELECT Function: This function is used to restore individual queue entries residing on tape to the VSE/POWER queues (RDR, LST, PUN or XMT) according to selection criteria. The selection criteria are specified by the operator in response to message 1R41D. Only queue entries with matching selection criteria are reloaded to spool. The queue entries will be restored according to their class and disposition.

The following figure illustrates the format of valid spool entries as they would appear on physical tape. This is true for both VSE/POWER spooling (DISP=T) and POFFLOAD tapes. If using a 9346 or 3592 tape unit, then the tape may contain an invalid spool entry with a missing trailer queue record which is ignorred.

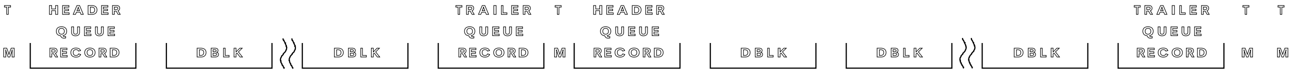


Figure 28. Internal Tape Format

Refer to “Queue Record Area (QRA)” on page 614 for the layout of the header and trailer queue record and to “Logical Data Record Area (LDA)” on page 539 for the layout of each DBLK.

BACKUPnn/SAVEnn/PICKUPnn Functions: Performs the BACKUP or SAVE or PICKUP function, however the queue record saved to tape is in the (length) format of the indicated 'nn' release, allowing the user to perform a LOAD or SELECT queue entry(s) on a down-level 'nn' system.

Dynamic Partition Support

Coexisting with the support for spooled static partitions, VSE/POWER offers to activate the dynamic partition scheduling task using the PLOAD DYNC command, which also loads and activates a user defined table, called Dynamic Class Table. It contains characteristics of job classes, for which dynamic partitions may be allocated, that process jobs of the corresponding job input class.

The dynamic partition scheduling task (DPST - with code in IPW$$DP) operates as a static 'hyper'-execution reader task and selects a reader queue entry for - let us assume - class Q; however no partition to process the job is available yet. Hence the dynamic partition scheduling task allocates a partition/address space of e.g. type Q and starts an execution reader task for e.g. partition Q1, passing the selected job entry directly to the newly started task.

Job Control residing in e.g. partition Q1 asking for the first and further job statements is then serviced by the execution reader task, as if it was a static partition. Only when the end of the VSE/POWER job has been reached, the dynamic attribute becomes visible again: Job Control is informed to do partition clean up and thereafter the Q1 execution reader task de-allocates the whole partition/address space Q1 and detaches itself. So this task is alive only for the processing period of one VSE/POWER job and has to be revived by the permanent dynamic partition scheduling task when required.

Any dynamic class may at the same time also be a job input class of a static partition; in this case both static and dynamic partitions compete in processing a job of this class. In the past it has been the user's responsibility to assign input classes to the static partitions according to job and partition profile. With the introduction of dynamic partitions, the user is given both full freedom and responsibility to decide, which job class should be used as a dynamic and/or a static one.

For the dynamic partitions the partition control block is allocated in system GETVIS area during dynamic partition allocation. The device entry list has space for the number of spool devices defined in the dynamic class table.

Enabling Dynamic Partition Scheduling

Already at VSE/POWER initialization time, the dynamic partition scheduling task is attached. The task is equipped with a dynamic partition control block (called DPCB), introduced as a new VSE/POWER resource to control parallel accesses to the list of dynamic class-table-pointers. Next the dynamic partition scheduling task waits for being posted by

- PEND termination, from where on dynamic partition scheduling will be deactivated
- or by the first PLOAD DYNC command issued.

When the operator desires to activate dynamic partition scheduling, he may enter the PLOAD DYNC,ID=x,FORCE command, which results in following action steps:

- load the VSE/AF library member 'DTR$DYNx.Z' into the VSE/POWER area using VSE/AF service, (DYNCLASS ID=GET) and run below verification for each class:
  - identify 'invalid' flags set by VSE/AF consistency checking of class characteristics
  - pre-check definitions of reader, printers and punch devices to be spooled for existence and validity of device type, and set 'invalid' flags in case of failure
- move the loaded Dynamic Class Table into the Supervisor area as **active** Dynamic Class Table using VSE/AF service (DYNCLASS ID=LOAD)
- set all dynamic classes **enabled** using VSE/AF service, (DYNCLASS ID=LOAD) provided they were not flagged 'invalid' during the verification process
- produce a status display of the active Dynamic Class Table on the console or as a list queue entry

- collect all enabled classes from the Dynamic Class Table and set up the corresponding VSE/POWER reader queue class table pointers in the DPCB, so that the dynamic partition scheduling task may find dispatchable jobs in these reader queue classes.
- inform the dynamic partition scheduling task to take over.

Driving Dynamic Partitions

Selection by Dynamic Partition Scheduling Task (DPST): Operating as a 'hyper'-execution reader, looking for jobs - not only in four but in up to ten classes, the dynamic partition scheduling task enters the VSE/POWER Get-Next-Job function, after it has been activated by the first PLOAD DYNC command.
When no selectable job can be found, the task returns to wait for the following events:

- PEND termination
- Add-To-Queue of a reader job
- PVARY DYNC modification of any 'enabled' class state
- PLOAD DYNC replacement of the active Dynamic Class Table
- any partition de-allocation has been done at end-of-job time

When a job is found, the Get-Next-Job function sets it 'in execution' (DISP=*) state.

Start Partition by Dynamic Partition Scheduling Task (DPST): Using VSE/AF service ALLOCATE (passing 'class') the dynamic partition scheduling task triggers space allocation for a partition of the requested class type.
In case the ALLOCATE service returns:

- no more partition available in requested class
- or no more allocation space at all

the 'in execution' job is returned to the reader queue with its original disposition and possible due date, and the corresponding class is set 'suspended' for the further Get-Next-Job attempts. Then the dynamic partition scheduling task continues to look for more dispatchable jobs in the remaining classes using the Get-Next-Job function.
In case the ALLOCATE service returns:

- no more partition available at all

the 'in execution' job is also returned with original disposition, but all classes are set 'suspended', and the dynamic partition scheduling task waits for any partition de-allocation or another PLOAD/PVARY or the PEND command.

When allocation of partition space and Supervisor control blocks has been successful, the dynamic partition scheduling task obtains VSE/AF Partition Control Block Extension (PCE) information, which offers the SYSLOG-id of the new partition, e.g. 'Q1'. Then, by the internally issued command as e.g.

```
PSTART Q1,Q,A,,NPC (input class Q, default output class A (unless
                   SET DYNOUTCL=DYNCL), and 'No Priority Check'
                   versus priority of VSE/POWER partition)
```

the dynamic partition is activated, while the selected reader queue entry and the PCE address are passed directly to the execution reader task of the new partition. The VSE/POWER PSTART partition processor respects the following actions for dynamic partitions:

- obtain devices to be spooled from the corresponding class table entry
- bypass spooled device verification (already done in advance)
- anchor pointer to the VSE/AF information (PCE) into the VSE/POWER partition control block

- set up 'E' stop code for the execution reader task from the very beginning, to ensure processing of **one** VSE/POWER job only
- pass the selected reader queue entry to the newly created execution reader task for e.g. partition Q1, and identify this task as 'dynamic'
- activate the dynamic partition (as for static ones) by the existing VSE/AF service: TREADY COND=START
- inform the waiting dynamic partition scheduling task to continue for PSTART done.

  Note: This service sets the partition running and prompts Job Control to do dynamic partition preparation asynchronously, while the service itself returns to the caller immediately.

Whereupon the scheduling task returns to the Get-Next-Job function, to look for more dispatchable entries.

The Execution Reader for a Dynamic Partition: The dynamic partition *preparation* done by Job Control comprises allocation of partition related areas, assignment of devices, and execution of the 'profile' procedure defined in the class table. Then, as for static partitions, Job Control issues a spooled read request, to obtain the first statement of the job.

Thereupon the VSE/POWER execution reader task is given control. It resets the VSE/AF 'initialization-active' state, it bypasses the Get-Next-Job call and uses the pre-selected job entry instead, from where statements are passed to Job Control. After a possibly final '/&' has been transferred, VSE/POWER expects Job Control to continue reading (as with static partitions), so that the VSE/POWER end-of-job condition may be found. Being initialized with the 'E' stop code, the dynamic execution reader task enters partition termination processing and acts as follows:

- simulate read I/O completion with the 'do-cleanup' indication set for Job Control and wait for posting of the do-deallocation-ECB. Job Control cleanup processing for dynamic partitions comprises:
  - return partition resources
  - de-activate partition by VSE/AF service TSTOP COND=UNBATCH
  - post VSE/POWER to do partition de-allocation
- reset spooling indication of intercepted devices
- do not terminate partition with 'EOJ' code as with static partition but de-allocate the dynamic partition using VSE/AF service ALLOCATE (passing PCE-'PIK') and respect service return codes
- upon successful de-allocation event, the dynamic partition scheduling task is informed to resume all 'suspended' classes, because a new allocation attempt might now be successful again
- bypass existing message 1Q33I (STOPPED partition-id)
- return VSE/POWER resources and detach itself.

Note: Already dynamic partition preparation might fail due to e.g. 'devices not assignable'. Then Job Control will also inform VSE/POWER by the do-deallocation-ECB. So the dynamic partition execution reader task should wait for both events:

- first spooled read I/O request
- do-deallocation-ECB posted

In the latter case, the execution reader task will act as follows:

- issue msg 1Q6FI
- return job-entry to reader queue with original disposition and possible due date
- disable corresponding dynamic class using VSE/AF service DYNCLASS ID=DISABLE
- reset spooling indication of devices
- request de-allocation of partition
- return VSE/POWER resources and detach itself

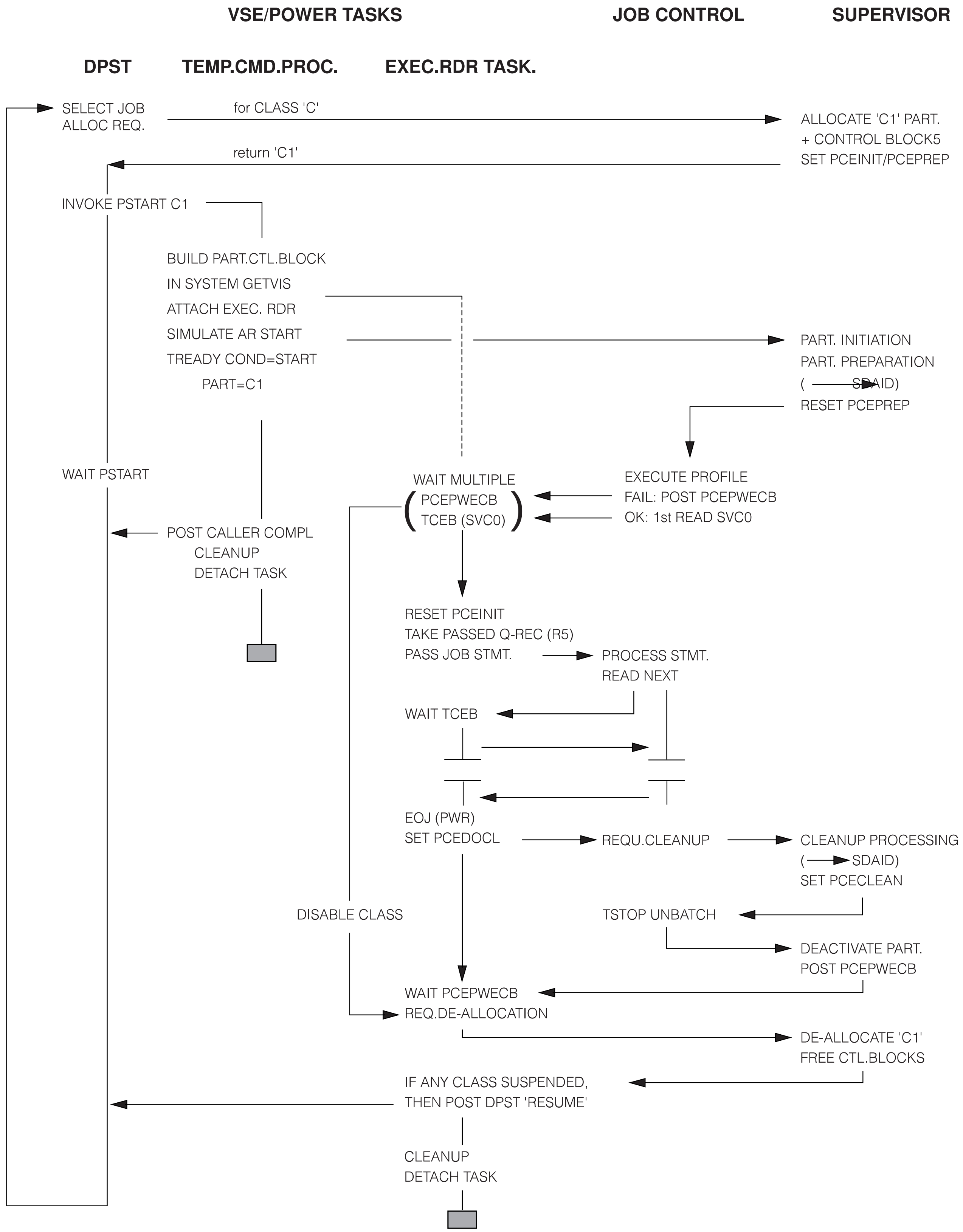


Figure 29. Flow of a Dynamic Partition

Tracking Dynamic Partition Allocation

No message is issued, when allocation of a dynamic partition fails because of maximum number of active partitions per class reached, since this limit is user specified. Allocation is resumed, as soon as any partition of the class terminates.
One of the various versions of message 1Q3FI is issued periodically, when allocation of a dynamic partition fails due to the reasons summarized in the "recovery table" of the module header of IPW$$DP.
To provide a system tuning aid for resources consumed by allocation of dynamic partitions, several statistic counts are maintained recording all events of both successful and failing allocation. Refer to the group of:

```
DYNAMIC PARTITION SCHEDULING STATISTICS
```

of the statistics example provided by the manual: *VSE/POWER Administration and Operation*.

Attributes and Restrictions

Selection of Dynamic versus Static Partitions: The current implementation yields the following behavior: Allocation of a dynamic partition is preferred versus passing a job to static partition for processing. That means, when a job for e.g. class C enters the system while both a dynamic class C is enabled and a static partition started also for class C is in waiting-for-work state, then the job will be started in a dynamic partition of class C. This can be changed if the SET DYNAL=LOW autostart statement is used. With this statement the dynamic partition scheduling task has the same dispatching priority as a local reader task.

VSE/AF Priority Relationship: VSE/POWER starts any dynamic partition internally with the 'NPC' operand of the PSTART command. Therefore dynamic partitions may have an arbitrary partition priority with respect to the VSE/POWER partition. Consequently the user takes full responsibility when assigning to a dynamic class a higher priority than to the VSE/POWER partition; that might deteriorate the performance of VSE/POWER own functions as networking, remote job entry, tape processing, and writer tasks.

VSE/AF Priority Reflection: Static partitions, once started, never loose their resources. However dynamic partitions compete dynamically in the common resources as address space and number of active partitions.
The VSE/AF PRTY command usually takes influence on the VSE/AF dispatching mechanism only. For dynamic partitions, where VSE/POWER triggers partition allocation, the operator determined PRTY is also reflected in the VSE/POWER dynamic partition scheduling mechanism:

- At every PLOAD DYNC or PVARY DYNC,ENAB event the sequence of dynamic class table pointers taking influence on the Get-Next-Job selection algorithm is re-ordered according to the existing PRTY sequence.
- Every priority change introduced by a new PRTY command will inform VSE/POWER by setting a PCE PRTY-change-flag. Whenever the dynamic partition scheduling task is about to (re-) enter the Get-Next-Job function and finds the change-flag set, the sequence of the class table pointers is re-adjusted immediately.

A group of priority balanced dynamic classes, for example C=D=E, is not reflected as such by VSE/POWER. Instead these classes are interpreted as an ascending priority sequence.

Spooling Restrictions and Defaults: The class table definition and verification means will guarantee, that dynamic partitions always have spooled reader, printers or punches defined; that means VSE/POWER will

- not support 'reader only' dynamic partitions
- not support 'writer only' dynamic partitions
- not support 'MT'-type dynamic partitions

Restrictions and Extensions to Existing Commands and Functions

- PALTER PARTITION,CLASS providing a new input class is not supported for dynamic partitions - any attempt is rejected by message 1R52I.
- PSTART PARTITION,X this command will always try to start a static partition - there is no operator means to start a dynamic partition. Any attempt is rejected by message 1R90I.
- PDISPLAY A,PART displays the currently active reader and writer tasks of all partitions on the central operator console, while 'PDISPLAY A,PART,partition-id' shows active tasks belonging to the specified partition. The 'PART' selection parameter may now also be:
  - 'SPART' ... to display tasks of all static partitions
  - 'DPART' ... to display tasks of all dynamic partitions.

  The 'partition-id' selection parameter may now also be:
  - 'dclass' ... to display tasks of all dynamic partitions belonging to the dynamic class 'dclass'
  - SYSLOG-id of a specific dynamic partition.
- PSTOP PARTITION operator initiated partition stop does not become effective for dynamic partitions, since they are stopped anyhow after completion of the one job, they have been started for.
- PFLUSH PARTITION will also be supported for 'partition' = SYSLOG-id of a dynamic partition. Consequently the 'PCANCEL jobname' command may also be used for jobs running in a dynamic partition.
- PGO PARTITION when prompted by message 1Q57A or 1Q58A for tape mounting while spooling to tape, this command allows to set up continuation conditions. It will also support the SYSLOG-id of a dynamic partition.
- PAUSING OF PARTITIONS when after system breakdown VSE/POWER has been restarted with the SET NORUN=YES autostart option and disposition X has been assigned to jobs in execution at system failure, then autostarted static partitions finding any job eligible to run issue message 1Q36I and are set into VSE/AF '// PAUSE' mode, so that the processing sequence of jobs may be rearranged.

  Dynamic partitions however will not enter the PAUSE mode under the same conditions.
- PARTITION DEPENDENT SUBSTITUTION SLI inclusion from a VSE sublibrary will replace the leading '$$' of the membername by 'c$', in case the job executes in a dynamic partition. 'c' is the dynamic class character, that means the first character of the SYSLOG-id of the dynamic partition.

Load, Modify, and Display the Dynamic Partition Support

Load Dynamic Class Table The dynamic class table member DTR$DYNx.Z is loaded by VSE/AF service DYNCLASS ID=GET,CLASS=ALL,AREA=DCLTCLS into VSE/POWER virtual storage for spooled device checking, for display and for possibly even activation in Supervisor area. If no virtual storage is obtained, the PLOAD command is terminated with message 1QA7I. Member DTR$DYNx may vary in size due to optionally 1-23 class table entries, each of them of constant class entry length. The class table entries have a logical END indicator (CLEOTAB). The spooled device lists of each class entry are located at fields (CLSRDR/CLSPRT/CLSPUN). Not used spooled devices are passed as X'000000'. Expecting a maximum DTR$DYNx.Z, VSE/POWER reserves storage in length of 23 class table entries.

The requested storage is preceded by 23*29 bytes, so that in each class one error byte is provided for every reader/printer/punch device (1+14+14). PLOAD DYNC spooled device checking sets these error bytes and additional summary bits in the corresponding class table entry. This technique allows to display a VERIFY report by a totally different task (PDISPLAY DYNC called internally), which is even able to produce a summarized error report.

Since PLOAD DYNC,VERIFY may also be processed by an internal cmd processor task (invoked by SAS-CTL) and since the requested virt. storage is handed over to other tasks for a display and possibly subsequent release, the permanent cmd processor is always made the owner. The storage is obtained with WAIT=NO option, not to block up the permanent cmd processor (might be needed to clear such a situation).

When all classes are specified correctly, processing continues depending on the option specified in the PLOAD command.

If VERIFY, then the following is done:

1. set VERIFY request (DCL1VER) in DCLT header area for 1Q6BI3 DYNAMIC CLASS TABLE VERIFIED
2. display of *verified* DCLT by internal call of the PDISPLAY DYNC command.

If (COND and any-class-invalid) then the following is done:

1. set COND request with error (DCL1CER)
2. display of *verified* DCLT by internal call of the PDISPLAY DYNC command.

If COND request with all classes valid or the FORCE request with even invalid classes the following is done:

1. Lock DPCB (and keep lock until DCLT is activated and enabled, since PDISPLAY DYNC might be running at the same time, or the DPST or $$XRE task might operate on the class table pointers in parallel)
2. Set DPC1CHAN to warn DPST, that enabled classes might have changed
3. make verified DCLT to active DCLT by following request:

   ```
         DYNCLASS ID=LOAD,AREA=()
   ```

   with AREA pointing to DCLT within DCLT area
4. If (RF)-return-code > 0 then handle load failure
5. save reg. 1 returned address of active DCLT in DPCBACT
6. enable all valid dynamic classes by setting up the PVARY DYNC,ENABLE,ALL command in DCLT header area and invoke internal command processor by

   ```
        IPW$ICP REQ=POWER,PASS=NOLOCK
   ```

   requesting not to lock/unlock the DPCB, which is owned by the calling task itself
7. wait for $$CV command completion using IPW$WFC DPCBECBL
8. set DCL1CFOK for 1QB6I ...LOADED SUCCESSFULLY
9. Loop over all classes of active DCLT and check CLERFLG1:
   a. If any class invalid
      then set DCL1FER for 1QB6I1 ...LOADED - WITH INVALID CLASSES, reset DCL1CFOK again, and leave loop.
10. End loop over all classes
11. Unlock DPCB to free permanent command processor (and even temporary) as soon as possible, and trigger a display of the *active* DCLT by internal call of the PDISPLAY DYNC command.

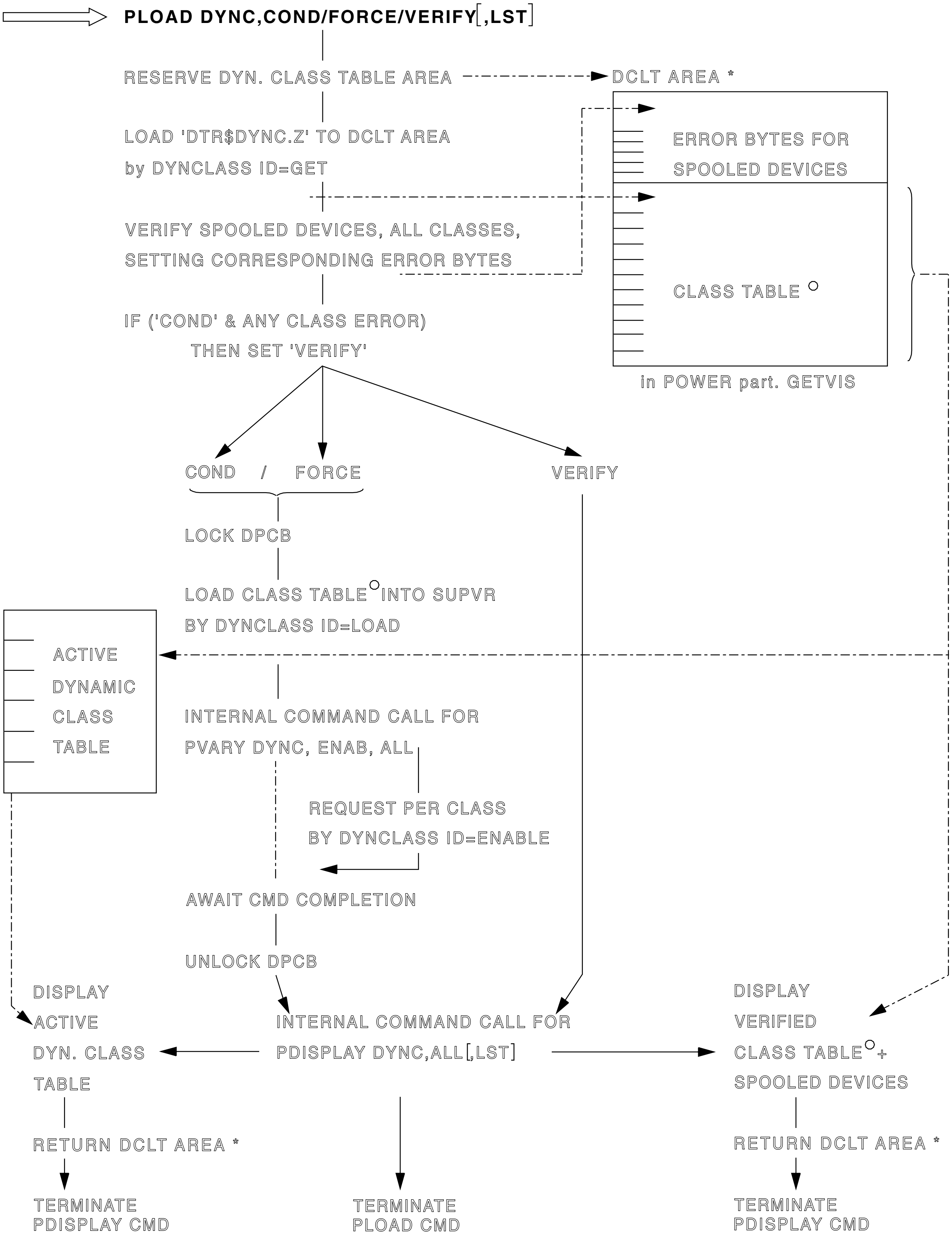


Figure 30. Logic Flow of PLOAD DYNC Command

Modify Dynamic Classes in a Running System: Whenever dynamic classes have once been enabled, they may be de-activated again by the PVARY DYNC,DISAB command (using VSE/AF service DYNCLASS ID=DISABLE). The dynamic partition scheduling task is informed to reduce its job searching algorithm by the just disabled classes. Any job still running in a dynamic partition belonging to a class disabled in between, is allowed to process till end-of-job.

Whenever dynamic classes have once been disabled or have been loaded with the 'initially-disable' specification, they may be enabled again by the PVARY DYNC,ENAB command using VSE/AF service (DYNCLASS ID=ENABLE). The dynamic partition scheduling task is informed to extend its job searching algorithm by the just enabled class(es).

Displaying Characteristics of Dynamic Classes: To get information about predefined features of various classes contained in the currently active Dynamic Class Table and to keep track of their enabled/disabled/invalid state at any time when desired, the PDISPLAY DYNC command is used. Characteristics of dynamic classes may be displayed on either console or may be collected in a list queue entry instead. The following selection parameters are offered for display:

- all classes
- enabled or disabled or invalid classes
- a selected class

Abnormal Termination with Dynamic Partitions

VSE/POWER Abnormal Termination: This condition is handled in the AB-exit of VSE/POWER. For all spooled partitions the following actions are taken:

1. set the do-cleanup indication for job control
2. cancel partition with code 'due to subsystem request'
3. force still outstanding I/O request complete
4. remove spooling indication of devices and unassign them
5. wait for cleanup completion (do-deallocation-ECB to be posted)
6. if dynamic partition then de-allocate the dynamic partition using VSE/AF service (ALLOCATE passing PCE-'PIK').

I/O Errors on Queue or Data File: For static and dynamic partitions the existing support handles

- READ/WRITE I/O error on Queue File, where the VSE/POWER V.2.3 repair methods make such an event transparent to the task suffering the error (see I/O error handler tables in IPW$$TR).
- WRITE I/O error on Data File, where the output 'in creation' is lost and a new output entry is started by segmentation.

Read I/O errors on Data File result in message 1Q6JI with job entry kept in hold disposition. Then the task terminator performs step 1)-5) as in the AB-exit. Finally dynamic partitions are de-allocated, while static partitions are dropped from VSE/POWER spooling control.

Failure of Dynamic Partition Support: If the interplay of VSE/POWER and VSE/AF Supervisor by internal services results in unexpected, illogical situations, VSE/POWER will state that by messages 1QB5I and 1QZ0I RC=0022 followed by an internal IDUMP requested to the DUMP sublibrary. VSE/POWER will ignore the failing dynamic partition request, and continue processing.

Interplay of Dynamic Partition Scheduling Functions

The following rules should be kept in mind:

1. PLOAD DYNC command establishes an active DCLT, calls PVARY DYNC,ENABLE to enable the classes.
2. PVARY DYNC,ENABLE enables classes in active DCLT and posts TCEB of DPST with DPC1ENDI indication set.
   Hence DPST sets up a new $WFM list of class table pointers for classes enabled in active DCLT and copies pointer list to suspended-ptr-area. DPST uses $WFM list for $GQS searching of next job and, if no job found, for $WFM waiting, to get informed about new job added to a class.
   When any partition allocation fails, the accessed class(es) must be suspended, namely by:
   a. - remove class ptr(s) from $WFM list, so that $GQS and $WFM functions only work for still remaining classes
   b. - flag class(es) (...making 'class' lowercase) in suspended-ptr-area, to be able and keep track of originally enabled classes.
3. When $$XRE does partition de-allocation and any class is found suspended in susp.-ptr-area, then TCEB of DPST is posted with DPC1RESU indication set.
   Hence DPST resumes all classes in suspended state (...making 'class' uppercase again), and copies original set of enabled classes from susp.-ptr-area to $WFM list.
   Advantage of keeping suspended-ptr-area:
   a. - after de-allocation only CONDITIONAL re-construction of $WFM list
   b. - fast re-construction of original $WFM list
   c. - tracking possibility of suspended/resumed classes
4. Alteration of VSE/PRTY of dynamic classes informs VSE/POWER by PCE flag 'PCEPRTYC' to adjust both its $WFM ptr. list and its susp.-ptr-area list, so that $GQS serves highest prty class first; DPST - whenever about to call $GQS - respects 'PCEPRTYC' setting.
5. PVARY DYNC,DISABLE disables classes in active DCLT and informs (...as ENABLE) DPST, by posting its TCEB with DPC1ENDI indication set. Thereupon DPST refreshes all its class table pointers.
6. Locking of DPCB is used to control parallel access to:
   a. - active DCLT (DPST, PLOAD, PVARY, PDISPLAY)
   b. - suspended-ptr-area (DPST, $$XR)

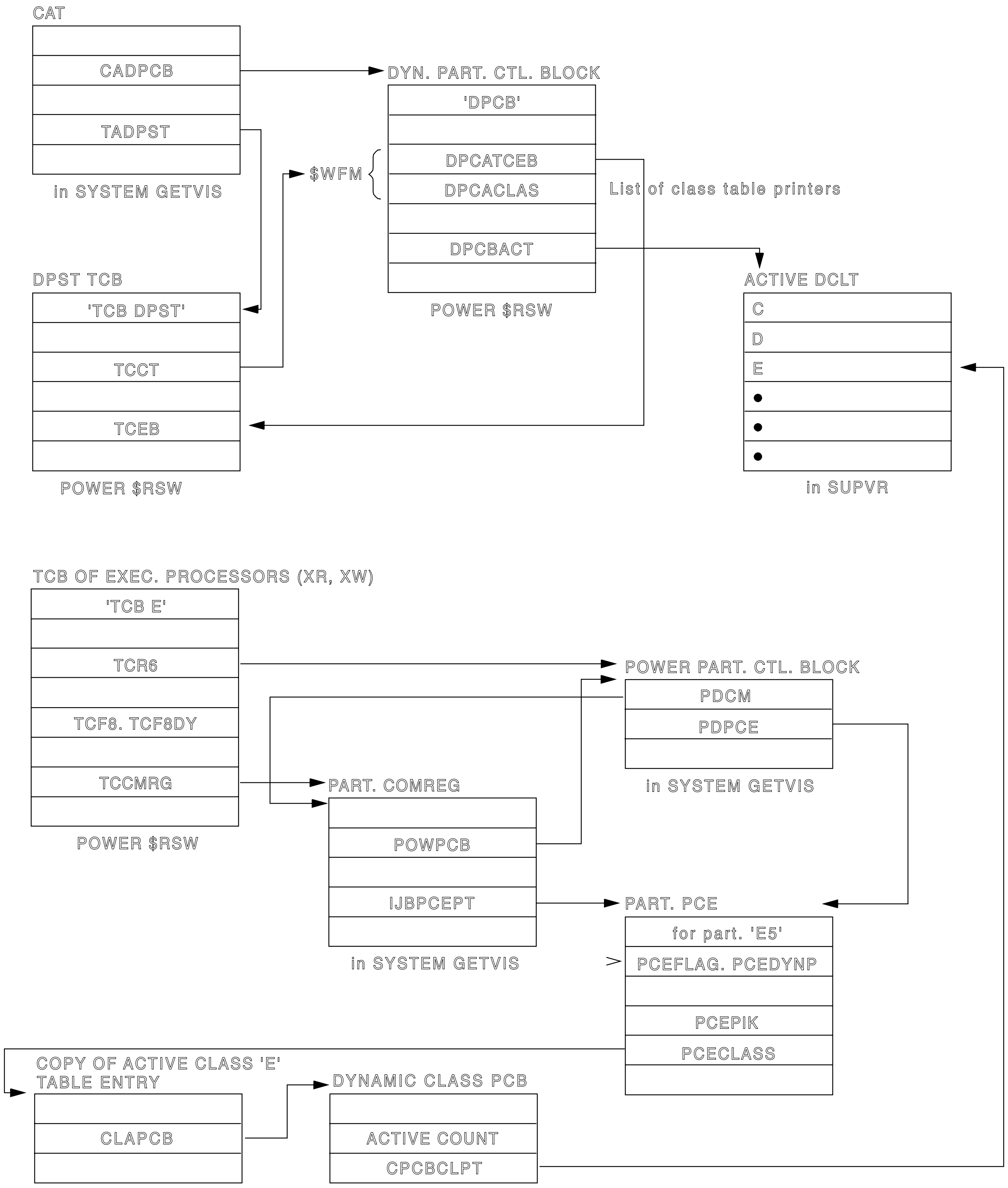


Figure 31. Control Block Relationship for Dynamic Partitions

The Spooling Process

Queue File Organization

The queue file consists of one or more tracks or FBA blocks. Logically, the queue file is organized in queue record blocks. The queue record block size depends on the device being used. The last queue record block is reserved for the master record. If the master record is larger than one queue record block, it occupies the last n queue record blocks. Each of the normal queue record blocks contains compartments. The compartment size is 384 bytes. A queue record block contains 32 compartments with each compartment containing one queue record of 368 bytes. If the queue file resides on a FBA device, the queue record block comprises 24 FBA blocks. The queue record blocks are addressed by their relative block number starting from 0.

Device Type	Block size	Number of Queue rec	Blocks/ track
3380	12288	32	3
3390 (as 3380)	12288	32	3
3390	12288	32	4
FBA	12288	32	—

Figure 32. Device Type - Queue Record Block Relationship

Four types of records are physically present on the queue file:

A master record — resides on the last "n" queue record blocks. The master record is the main control record on spool and contains the following information:

- Dispatchable class table
  Contains all queue entries "ready to run" with disposition D or K.
- Non-dispatchable class table
  This table holds all queue entries with disposition H, L or any temporary disposition A, X or Y.
- Node attached table
- Shared spooling control information
- Control fields which must be retained between the termination and initialization of VSE/POWER.
- Defect queue record block map
  This map consists of one bit for each of the 3125 queue record blocks to house a maximum of 100,000 queue records. The bit is on, if the queue record block is no longer accessible due to an I/O error previously occurred.
- Refresh table; the size of the refresh tab depends on the maximum number of queue records (currently 100,000) and the total number of sharing systems allowed in VSE/POWER. Currently this number is 9. The refresh table contains a flag for each block and each sharing system indicating if a 'refresh' of the queue record block is necessary before referencing any queue record contained in this block. In a non-shared environment, the refresh table is not used but present. A 2-byte field is used for each queue record block, whereby each of the last 9 bits of the 2-bytes represents a SYSID.

The master record is part of the Disk Management Block (DMB).

Queue records — queue identifier F, R, L, P or B

A dummy record	queue identifier D. Indicates the logical end of the queue records.
An internal record	queue identifier I (physically the first queue record in the first queue record block). Contains the relative queue record block number of the master record in its first four bytes.

Logically, by means of pointers (relative queue record number), the queue records are either a member of the free queue record chain or one of the various class chains. The free queue record chain is shown in Figure 33.

Records in the free queue record chain (queue identifier F) are chained by the next-in-set pointer. The start of this chain is kept in the master record.

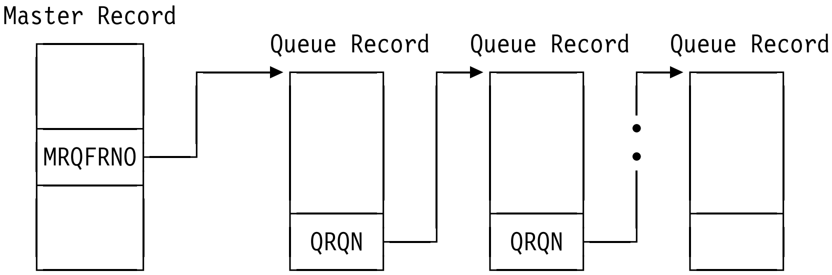


Figure 33. Free Queue Record Chain

For each queue entry of a class chain, there exists a queue record, which provides forward and backward pointers in the class chain (see Figure 34 on page 91). The queue record addressing format consists of a 4-byte field containing the relative queue record number.

One or more DBLK groups are associated with the queue entry. The various DBLK groups are linked together by a

1. forward chain via the Spool Environment Record (SER), that heads the last DBLK of each DBLK group
2. backward chain via the Spool Environment Header (SEH) record, that heads the first DBLK of each DBLK group

The queue record contains three control fields:

- Relative DBLK number of the first DBLK in the first DBLK group.
- Relative DBLK number of the first DBLK in the last DBLK group.
- Number of allocated DBLK groups.

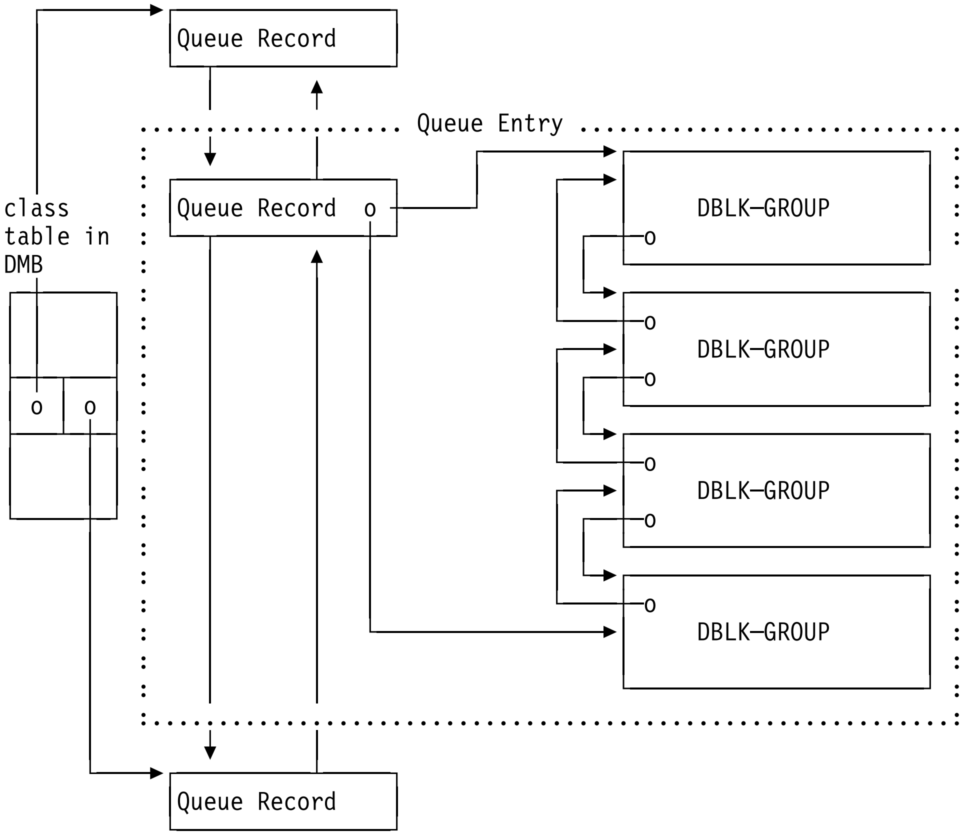


Figure 34. Class Chain and Queue Entry Structure

For more details on the logic structure of DLBK groups chained to a queue entry refer to VSE/POWER Administration and Operation, Appendix B, 'Analyzing dumps and traces'.

The master record and the queue record currently processed by a task are contained in storage. The master record is part of the Disk Management Block (DMB) and resides in fixed storage while the queue record resides in virtual storage.

In-Storage Queue Principles

Non-Shared Operation: At VSE/POWER start up time, all queue record blocks are read in and placed in VIO or partition GETVIS space -- building the storage copy of the queue file. Once the queue file is placed in storage, the following rules apply:

- All queue record read requests (via IPW$GQR macro) are fulfilled using the storage copy of the queue file.
- All class chain changes must be recorded via the IPW$MQR macro instruction; the update is only made in the storage copy of the queue file.
- All queue record status or attribute, such as disposition, changes must be recorded to the queue file disk copy, too, via the IPW$WQR macro instruction. This is because such changes must be visible after a breakdown of VSE/POWER.

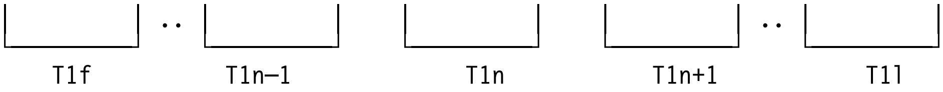
At VSE/POWER termination time, all queue record blocks as well as the master record are written back to disk, thereby committing all modifications made to queue records in the storage copy of the queue file.

When queue file recovery has to be invoked, which is done after all queue record blocks are read in, it examines each queue record using the sequential organization of the queue file and re-builds the various class chains and free queue record chain, thereby respecting the creation time stamp to reconstruct the class chain in the same sequence as before the breakdown. Queue file recovery uses the get queue record (IPW$GQR) service routine and queue management functions, thus ensuring that abnormal termination of the recovery process does not destroy the integrity of the queue file on disk. Only for the sake of a well arranged restart all queue record blocks are written back to disk at the end of queue file recovery.

Shared Operation: Each system maintains its own in-storage copy of the queue file. By reading and writing periodically from/to the shared queue file on disk, each system keeps an up-to-date copy of the queue file in its storage.

An overview of the shared spooling processing shows that there are two basic stages during a processing cycle. The first stage, called T1, begins with a lock of the queue file to ensure exclusive control of the queue. Next the master record is read. Then, using the refresh table in the master record, any queue record blocks changed by other systems are read. Once a system has control of the queue file, it is free to update it until the timer interval (T1) expires. During this stage, the system can read/write from/to the queue file without losing its ownership. When the timer interval ends, the timer task ensures that all changed queue record blocks are written back to disk, then unlocks the queue file, enabling the next system to start its T1 interval. The second stage is a dormant period during which a system makes no attempt to update the queue file and thereby allows the other system time to complete their active phase.

VSE/POWER provides a mechanism analogous to page-out. When a change is made to a queue record, the queue record block is marked as changed, so that it will be automatically written back to disk when the system finishes its T1 interval. Any change that is made to a queue record block is propagated to all other systems by means of the refresh flags.



Shared operation is characterized by T1 intervals. During this time the queue file is locked by the own system thereby preventing that another system updates the queue file.

- During the first T1(f) interval, the queue file is read into VIO or partition GETVIS space as in non-shared case.
- All follow on T1 intervals can be seen as a timely reduction of a non-shared operation, with a queue file refresh at the beginning of T1 and a modification commitment on disk at the end of T1. This is achieved by the refresh table, contained in the master record. The "write queue record" (IPW$WQR) service routine sets the refresh flags for all possible systems (SYSID's) in the refresh table associated with the queue record block being written. The "modify queue record" (IPW$MQR) service routine sets the "modified" flag, which is identical to the refresh flag for the own SYSID, in the refresh table when a queue record block is updated in the storage copy only. The "modify" flags are then examined at the end of the T1 interval and all queue record blocks flagged as such are written back to disk. Then the "refresh" flags for all other systems (SYSIDs) are set.
- During the last T1(l) interval, all modified queue record blocks are written back to disk and finally the master record is updated and written to disk by the timer task itself when the last T1 interval is indicated; this is done by setting the immediate termination code "S" in the TCB of the timer task.

If the VSE Access Control function is activated (security SEC=YES) during shared operation, then the scheduling of RDR job entries for the execution RDR task (IPW$NQS) can depend on the security characteristics of the job. If the job has inherited propagated security authorization from a parent job (i.e. "authenticated") and no explicit SYSID is indicated in the * $$ JOB card for execution, then the job will be selected to execute on that shared system which has a matching security zone (SECNODE) as the job. If no such CPU exists, then any CPU is allowed to execute the job.

Benefits in Case of Queue File I/O Errors

Read I/O errors: Since all queue record blocks are read in at initialization time, non-readable queue record blocks are flagged “bad” both in VIO or partition GETVIS space for later recovery and in the defect queue record block map in the master record. According to this flag all further read/write attempts are bypassed in order to isolate the erroneous disk area. The queue records of the “bad” queue record block(s) are lost, but queue file recovery rebuilds the class chains from the remaining queue record blocks - thus continuing with the subset of the queue file as long as desired.

Any master record read error results in a “master record reconstruction” attempt from generation values followed by queue file recovery in order to rebuild the various class chains and the free queue record chain. The operation may then continue at the loss of the free DBLK group subchains.

Write I/O errors: Whenever a queue record block or master record write fails, the most up-to-date copy of the queue file resides in storage (VIO or partition GETVIS space). That enables VSE/POWER to attempt a repair action by formatting the queue file (CKD only) followed by writing the storage copy back to disk. However, if the formatting or copying back to disk fails, VSE/POWER continues to operate with the in-storage copy and the data file on disk. The operator is then asked for soon POFFLOAD backup of the entire queue file. A cold start is required at next start up, since the queue file on disk no longer matches with the data file due to the in-storage operation.

Data File Organization

The space available on the data file is arranged in DBLK groups. A DBLK group is the smallest unit that can be allocated to a VSE/POWER job. Each DBLK group contains an integer number of DBLKs. The smallest DBLK group consists of one DBLK. The number of DBLKs comprising one DBLK group is defined at VSE/POWER generation time by means of the DBLKGP parameter. A DBLK group can cross track, cylinder or even extent boundaries. Logically, the entire data file is seen as a contiguous space, divided into DBLKs, whereby a certain number of DBLKs is grouped together and referred to as DBLK group.

Each DBLK is addressed by the relative DBLK number; the 31 bits of the relative DBLK number will allow a maximum of 2,147,483,647 DBLKs. The first DBLK has the relative DBLK number 0. The translation of the relative DBLK number to specific device geometries is performed using device specific information contained in the module control blocks associated with the data file.

Up to 32 data file extents can be specified, whereby all extents must reside on devices of the same type.

Each physical record (DBLK) in the data file contains one or more logical records. Each logical record represents a unique record of the user program that is being spooled. Figure 35 shows the layout of a physical record:

```
┌───────┬───────┬───────┬────────┬────────┬──────────┬─────────────┬──────────//────────────┐
│  RL   │  GP   │  CC   │  GP2   │  GP3   │  EL(R)   │  data *     │   binary zeros         │
├───────┴───────┴───────┴────────┴────────┴──────────┴─────────────┼──────────//────────────┤
│  2       1       1        1        1         2           n       │                        │
│                                                                  │                        │
│◄────────────── logical record ─────────────────────────────────►│                        │
│◄───────────────────────────────── logical data area ──────────────────────────────────────►│
```

RL   length of the logical record (including 8-bytes prefix)

GP   general purpose byte (see also sections “Logical Reader (IPW$$LR)” and “Logical Data Record Area (LDA)”)

CC   command code associated with the user channel program

GP2   general purpose byte 2

GP3   general purpose byte 3

EL(R) extended record residual length (for the first record extension, is equal to the total record length)

*   trailing blanks are suppressed

Figure 35. Data Record

Job Header, Job Trailer, and Data Set Header Records: Each queue entry, either job, list, or punch output, is preceded by a job header record and followed by a job trailer record. Each output queue entry (list or punch) has in addition a data set header record following the job header record. Data set headers may be present in a job queue entry to signify a change in record characteristics.

The job header, job trailer and data set header contain information which is required for job routing, execution, printing, punching and accounting.

Basic Header/Trailer Format: The basic organization of the job header or job trailer records is shown in Figure 36.

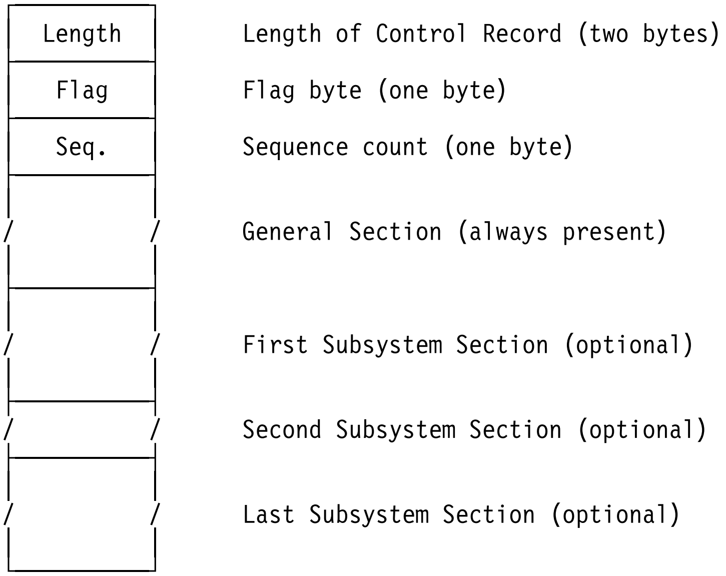

Length
Length of Control Record (two bytes)
Flag
Flag byte (one byte)
Seq.
Sequence count (one byte)
General Section (always present)
First Subsystem Section (optional)
Second Subsystem Section (optional)
Last Subsystem Section (optional)

Figure 36. Job Header, Data Set Header and Job Trailer Format

Job Header and Job Trailer: The job header record contains four types of information relating to the job as a whole:

- Identification (job name, job number, originator's name)
- Routing Control (execution node name, default print/punch node and remote names).
- Execution Control (job class, priority)
- Accounting Information (account number)

The job trailer contains accounting information dealing with the actual execution of the job (actual print line count, execution time, etc.).

The job header and job trailer records are built when a job first enters the system, or are already with the job if it is received from the network. The job header record and job trailer record may be updated with certain information by all systems on which, or through which, the job passes in the course of its transmission. Both job header and job trailer records accompany the job throughout its processing whether or not the job is processed locally or transmitted to another node.

These two records are built by the logical reader (IPW$$LR) when a job is first read in a VSE/POWER system. The job header record is constructed from the information in the * $$ JOB statement, if present, otherwise the VSE/POWER defaults are taken.

A job header record is also constructed, if a job is received via the network (IPW$$NR2) and the job is destined for the local reader queue (the receiving node is the final destination) and contains a * $$ JOB statement. The newly built job header record contains just few information out of the received job header record (for example origin node and userid). Most of the information is retrieved out of the * $$ JOB statement. If an operand is not specified in the * $$ JOB statement, VSE/POWER defaults values are used.

For more information concerning the fields in these two records, please refer to “Network Job Header Record (JHR)” on page 559.

Data Set Header: For each output created on a VSE/POWER system a data set header record is built from the information in the * $$ LST/PUN statement, if present, else from the defaults in the job header supplemented by the VSE/POWER defaults.

Every data set, list or punch, must be preceded by a data set header record. There may be multiple data set header records within the bounds of a job header and job trailer record.

Note: VSE/POWER does not create queue entries consisting of more than one data set, but they may be received from other systems within a network.

The data set header record describes the characteristics of the output and what to do with it. It contains three types of information:

- Identification (data set number)
- Routing Control (destination node name and remote)
- Data set characteristics (record format, record length, output class, printer setup information).

Queue File and Data File Processing

The queue and data file are maintained by the queue and data function routines:

- Reserve queue record (IPW$$RQ)
- Add queue entry (IPW$$AQ)
- Delete queue entry (IPW$$DQ)
- Free queue entry (IPW$$FQ)
- Put data record (IPW$$PD)
- Queue management services (IPW$$SQ & IPW$$Q1)
- Data management services (IPW$$DS)

Retrieval on the queue and data files is performed by the function routines:

- Get next queue entry (IPW$$NQ)
- Get data record (IPW$$GD)

Reserve Queue Record Function (IPW$$RQ): The reserve queue record routine is invoked via macro IPW$RQS. The routine locks the DMB for the duration of the processing and obtains then the first queue record from the free queue record chain, addressed by the master record and allocates this queue record to the calling task. If no free queue record exists, message 1QF4I is issued and the task then waits until a free queue record becomes available, unless the 'no wait' flag is set in which case immediate return is made to the caller with appropriate return code set. Otherwise a test is made to determine if the number of allocated queue records exceeds the value, defined by the SPLIM parameter. If so, message 1QF0I is issued if the message has not been issued for at least 60 seconds.

If a free queue record was available, a DBLK group is allocated to the queue record by means of the IPW$IQS REQ=ALLOCGP macro instruction. On return, the assigned DBLK number is stored in the queue record and the number of used DBLK groups is set to one. If the DBLK group allocation routine indicates that no free DBLK group is available (only done when the 'no wait' flag is set), the already allocated queue record is returned to the pool of free queue records, the various counters are updated accordingly, the DMB is unlocked and return is made to the caller with an appropriate return code. Finally, the queue record is written back to disk by means of the IPW$WQR macro instruction.

Add Queue Entry Function (IPW$$AQ): The add queue entry function is invoked via the IPW$AQS macro instruction. The queue entry, addressed by the calling task, is added to the appropriate class chain. If the disposition of the queue entry is D or K, the queue entry is added into the dispatchable class chain; if the disposition is L or H, or the queue entry is marked appendable (disposition A), or held due to output processing failed (disposition Y) or abended (disposition X), the queue entry is added into the non-dispatchable class chain. The queueing is performed based on priority whereby output queue entries with the same forms identifier are grouped together. If queue file recovery is active, the queueing is performed in time stamp sequence (the time stamp is saved in the queue record when the queue entry is added to the class chain). The position of the queue entry in the class chain is determined by stepping through the class chain in reverse sequence starting from the tail. If the 'keep' option was specified, the queue entry, which is already added in one of the various class chains, is only written back to disk by means of the IPW$WQR macro instruction.

If the execution node for a job (that is the system upon which the job is to be executed) or the destination node for list or punch output is not the local node, the job or output is placed in the transmission queue (XMT). This queue consists of two entries: One for jobs and one for output. The queueing is performed on a priority base only.

The output-available flag is set in the line control block or logical unit control block (LUCB) when the output is destined for a remote workstation that is currently logged on (terminal or logical unit).

When a job or output is destined for processing by a specific SYSID in a shared-spooling environment, the SYSID class table is updated depending upon the specification of the class and SYSID.

If a job or output is destined for another node in the network, a check is made if a connection exists to the prime route node. If so the first inactive transmitter found is posted. If no such connection exists, a check is made if a connection exists to the alternate route node. If so, the first inactive transmitter found is posted.

If the destination of the job or output is unknown, the local operator, and the originator if Notify was specified, are informed via message 1RA1I and the job or output is put in hold status in the XMT queue.

If the output is destined for an external device (that means the external device is eligible to process the output queue entry), and the external device is waiting for work, an output arrived signal control record is built and added at the tail of the order queue anchored to the external device control block (EDCB) associated with the device. The device service task is then posted to forward the signal.

When running in a shared spooling environment and a 'to' user id is specified for the queue entry being added, the QCA is scanned and each 'waiting for work' slot, which fulfills the criteria (matching destination, class, queue type, and system identifier) is posted.

To process the jobs with time event scheduling information, the following is done:

1. Before adding a dispatchable job with time event scheduling information which is destined for the local reader queue, the next due date is calculated by calling the module IPW$$TQ. The interface macro IPW$ITQ with the ADD operand is used for this purpose. When returning from IPW$$TQ the entry is processed according to the information in the queue record:
   a. if QRDG0W = OFF, the due date has expired and the queue entry is queued as today (according to its class and priority).
   b. if QRDG0W = ON, the due date has not yet expired and the queue entry is queued as the last queue entry of the dispatchable class chain.
2. If the due date of an already queued job (of the 'wait for run' subqueue) has expired, IPW$$AQ gets called from the TES task (IPW$$TV). In this case (TCTI="YTES") the above described call to IPW$$TQ is not done, because the TES task itself decided already, if the job is now eligible for immediate scheduling or not. A new jobnumber is assigned, if the job runs more than once, recovery is not running and the job already executed once. The job is queued as today (according to its class and priority).
3. Avoiding posting of a shared system, if the entry is chained into the 'wait for run' subqueue.
4. Avoiding posting of class table, if the entry is chained into the 'wait for run' subqueue.

Delete Queue Entry Function (IPW$$DQ): The function routine is invoked via macro IPW$DQS. The routine unchains the queue entry, identified by the queue record contained in the queue record space addressed by the requesting task's TCB, from its class chain. If purge of the queue entry was requested (QRDI=P), the queue record disposition is forced to D. The disposition attribute of the queue record is then examined. If the disposition is keep "K", it is reset to leave "L" and the queue entry is added to the non-dispatchable class chain by means of the IPW$AQS macro instruction. If the disposition is not kept the queue entry is marked as incomplete. This causes that a subsequent IPW$FQS macro call frees the queue entry.

If the 'hold' option was specified, the queue entry is not unchained from its class chain; instead the queue record is written back to disk by means of the IPW$WQR macro instruction with the original disposition.

To process jobs with time event scheduling information, the following is done:

1. If the job is in the 'wait for run' subqueue, it is removed from the 'wait for run' subqueue by branching to module IPW$$TQ using the interface macro IPW$ITQ with the option DEL.
2. If the job has been executed in a partition, the macros IPW$DQS and IPW$FQS are issued by the caller (IPW$$XRE) in this sequence to perform the deleting of the queue entry (if disposition D) or the requeuing of the queue entry (if disposition K) into the non-dispatchable class chain. This requeuing is done in IPW$$DQ which sets the disposition K to L and issues IPW$AQS (nested call of IPW$$QM !). Therefore if a job with a due date should run more than once, the disposition is not changed from K to L and the job is added to the dispatchable class chain once more. Such a job is not freed later on by IPW$FQS.
3. If the LOCATE request has been issued by the TES task, the indication 'do not delete' is set, which is used by the recovery module IPW$$RY to decide whether the entry should be deleted or added again to the queue file after an abnormal termination of VSE/POWER occurred during the delete process of this queue entry.

Free Queue Entry Function (IPW$$FQ): The function routine is called via macro IPW$FQS. The queue entry represented by the queue record contained in the calling tasks's queue record area is to be freed:

- if the queue record is not yet in the DELetion queue, it is added to the DELetion queue by IPW$WQR. When no more browser is active for this queue entry, the init./termination task is posted for 'final freeing' of the queue entry by another call of IPW$FQS.
- if the queue record is already in the DELetion queue and no more browser is found active, the queue record is written (IPW$WQR) free and added to the free queue record chain. All DBLK groups belonging to the queue entry are returned to the free DBLK group subchains by means of the IPW$IQS REQ=FREEGPS,LOCK=TEMP request, which tolerates temporary UNLOCK/LOCK of the DMB as long as SER-DBLK's are written free, because the calling tasks are well known.

If the queue entry is destined for the local reader queue, the disposition is 'K' and time event scheduling information for more than one run exists, the queue entry is not freed.

For details see “Access to Active Queue Entry” on page 101.

Put Data Record Function (IPW$$PD): The function routine is invoked via macro IPW$PDR. The routine moves a logical record into the data block (DBLK), also referred to as logical data area. If necessary, the trailing blanks of the logical record are suppressed and the shortened record is moved to the DBLK, if there is enough space. The page, line and record counts are updated accordingly. The spool environment block (SPB), containing among others the current printer set up, is updated when the just spooled record is a DSHR, select TRC or execute order record.

If the output remainder is not large enough to contain this logical record, the last (previous) record in the DBLK is made an 'end of block' record and the DBLK is written to the data file as a physical record. If the logical data record is a 'end of data' record, the DBLK is also written to the data file. After the DBLK is written, the current DBLK number is incremented to address the next DBLK in the group. If, however, the current DBLK is the last DBLK in the current group, the Spool Environment Record (SER) is completed with the various account values, such as page, line and record counts, and information extracted from the SPB and the DBLK is written to the data file. If the current DBLK is the first DBLK in a group, the Spool Environment Header (SEH) record is completed with the same spooling account values as recorded in the last SER record. The SER or SEH is used to facilitate restarting of output queue entries. A new DBLK group is then allocated by means of the IPW$IQS REQ=ALLOCGP macro instruction and the number of allocated DBLK groups is updated accordingly.

If the data record is larger than the DBLK size - 8 bytes, the record is split into record extensions that fit in the data block. These records are called extended records.

For each queue entry the current position is counted in new fields QRCCNR, QRCCLC and QRCCPG to prepare record and page counters used by “Get In Creation Queue Entry” on page 102.

Queue Management Services (IPW$$Q1): Queue management services consist of the following set of routines:

- DBLK group allocation
- DBLK group deallocation
- Queue file formatting

DBLK Group Allocation: A new DBLK group is allocated by means of the IPW$IQS REQ=ALLOCGP macro instruction. The service routine examines the free DBLK group subchains for a free DBLK group. If a DBLK group was found, the 'head' chain pointer in the master record is updated to address the next free DBLK group, the number of free DBLK groups is decreased and the master record is written back. The SER DBLK of the acquired group is also written to the data file marking the group no longer 'free' but used and owned by the 'queue record number' of the caller's queue entry. The DBLK group is allocated from the subchain containing the most free DBLK groups. Finally, a test is made to determine if the number of allocated DBLK groups exceeds the value defined by the SPLIM parameter. If so, message 1QF0I is issued, if the message has not been issued for at least 60 seconds.

The DMB is locked by means of the IPW$RSR macro instruction during the processing of the allocation function. If no free DBLK group is found, the function issues message 1Q38A NO DASD SPACE AVAILABLE FOR task, cuu and releases the DMB and waits. Execution writer tasks use IPW$WFM for the Data File ECB and the SEGMENT/CANCEL ECB, for which TCVEB is reused. Other tasks use also IPW$WFM, although the ECB list contains only the Data File ECB.

DBLK Group Deallocation: DBLK groups are replenished in a free DBLK group subchain by means of the IPW$IQS REQ=FREEGPS macro instruction. The routine issues the IPW$RSR macro instruction to lock the queue file for exclusive use. The DBLK groups addressed by the calling task in registers 1 and 2 are then added at the top of the free DBLK group subchain. The SER DBLK of each returned group is written to the data file marking the group no longer 'used' but 'free'. The DBLK groups are added to the subchain containing the lowest number of free DBLK groups. Next, the routine posts the event control block, indicating that at least one free DBLK group exists, writes back the master record and unlocks the queue file by means of the IPW$RLR macro instruction.

Queue file formatting: This routine is entered at cold start time or when an I/O error occurred on the queue file and the queue file should be re-formatted. The routine is invoked by means of the IPW$IQS REQ=FORMAT macro instruction. The routine formats the queue file by writing count fields on the track (CKD only). One entire track is formatted with one I/O. Any I/O error detected is reported back to the calling routine.

Get Next Queue Entry Function (IPW$$NQ): The function is invoked via the macro IPW$NQS instruction and acts according its invocation (see below). The routine returns the queue record of the first eligible queue entry belonging to a class chain referenced by the calling task. Class chains related to the calling task are identified by a task class list in the TCB. Each entry in the task class list is examined in turn. If the class chain it addresses is not empty, the class chain is stepped through until a queue entry is found which is eligible for processing. Typically, only the dispatchable class chains are examined and queue entries already being 'active' (DISP=*) are not selected by the Get Next Function.

- If the requesting task is a spool access service (SAS) task, the queue entries of the specified class chains are scanned for a queue entry with matching jobname, number and job suffix, if specified. If no such queue entry is found, or the queue entry is either protected or already in use, appropriate return codes are set in the function return code byte of the TCB which is translated to a return code/feedback code passed to the SAS application program.
- If the requesting task is a Job|Output transmitter, each queue entry in the Job respectively Output chain of the XMT queue is examined to see if it is eligible. The queue entry is eligible when the calling transmitter is sending to the prime route node or, if no connection is established to the prime route node, the transmitter is sending to the alternate route node. Again only the dispatchable job/output chains are examined and queue entries already 'being processed' (DISP=*) are never selected by this Get Next Function.
- If a **direct** SAS GET task requests a queue entry, it is accessed by its internal queue record number passed by the requestor and without searching through the class chains. The queue entry must be dispatchable and inactive (DISP=D|K) and all further specified queue entry attributes must match. If no such queue entry is found, or the queue entry is either protected or already in use, the requestor is informed by the function return code byte of the TCB which is translated to a return code/feedback code/feedback2 code passed to the SAS application program.
- If a SAS GET BROWSE task requests a queue entry, each entry in the task class list is examined in turn. If the class chain it addresses is not empty, the class chain is stepped through until a queue entry is found which is eligible for processing. Dispatchable and non-dispatchable class chains are examined and even queue entries already 'being processed' (DISP=*) are selectable by the Get Next Function. More than 1 SAS GET BROWSE task may select the same queue entry, which is additionally controlled by the multiple access count (MACC). After checking the MACC, the queue entry attributes are checked against the specification of the requestor and if no such queue entry is found or the queue entry is protected, appropriate return codes are set in the function return code byte of the TCB which is translated to a return code/feedback code passed to the SAS application program.
- A **direct** SAS GET BROWSE request combines the direct access method of the direct SAS GET request and the removed disposition limitation of SAS GET BROWSE.

Queue access type specifications: The queue entry passed back to the caller of IPW$NQS is set to 'active', except for (direct) SAS GET BROWSE, which does only change (increase) MACC. All calling tasks setting the queue entry to 'active' are called UPDATE tasks, SAS GET BROWSE tasks are now also called BROWSE tasks and the command processor task, which uses its own queue selection routine and **not** IPW$$NQ, is called a COMMAND task. A MODIFY task is a SAS PUT task selecting an already completed queue entry for PUT OPEN APPEND or PUT OPEN RESTART. It removes the queue entry from its chain, so that it looks similar to a CREATE task, which is any task creating a **new** queue entry.

Overview concurrent queue entry access limitations

current task processing a queue entry	access to entry requested by another task allowed: Yes/No ; Restricted by Resource/flag			
	COMMAND	UPDATE	MODIFY	BROWSE
COMMAND	No ; DMB	No ; DMB	No ; DMB	No ; DMB
UPDATE	No ; '*'	No ; '*'	No ; '*'	YES; MACC
MODIFY	No ; QRQP	No ; QRQP	No ; QRQP	No ; QRQP
BROWSE	Yes;	Yes;	No ; MACC	Yes; MACC
CREATE (note)	No ; QRQP	No ; QRQP	No ; QRQP	No ; QRQP

The MACC limit for parallel browsing remains the same (255 for non-shared or 15 per shared-spooling SYSID).

Note: Output entries created by execution processors may be segmented by command and may be browsed.

If all addressed class chains are empty or contain non-dispatchable queue entries, the routine releases the queue record area, clears the address of the queue record area, and returns to the caller. The calling task is then either placed in a wait state until a new queue entry is added or an existing queue entry becomes dispatchable or detaches itself. For a GETSPOOL request, the queue records are scanned for a matching job name, password, and class with the one supplied in the GETSPOOL SPL. In a shared spooling environment, the class entry that is associated with the SYSID is reset when no queue entry is found that is eligible for processing.

Only queue entries where the 'to' user id matches one of the logical destinations associated with the device service task are eligible for processing when the calling task is a device service task.

To process jobs with time event scheduling information, the following is considered:

1. A queue record of the 'wait for run' subqueue must not be returned to an execution reader task.
2. An entry of the local reader queue with disposition D or K and time event scheduling information is returned to a cross partition task (SAS, DST) using a GET request only if the BROWSE option is used.
3. An entry of the local reader queue with disposition D or K and time event scheduling information is not returned to a task which uses the old GETSPOOL interface.
4. An entry of the local reader queue with disposition D or K and time event scheduling information must not be returned to a POFFLOAD task.

If the caller is the POFFLOAD PICKUP function, only eligible entries are returned to the caller which have been tagged with the PICKUP flag indicating that the entry is to be backed up to tape. Both the dispatchable and non-dispatchable class chains are scanned.

Access to Active Queue Entry: Because a (direct) SAS GET BROWSE request does no longer set a browsed queue entry to 'active' state, even an 'active' queue entry may be selected for processing by a BROWSE task. The concurrently accessing UPDATE task **may delete** the queue entry when it finishes its processing while the SAS GET BROWSE task is still reading the queue entry.

To protect the SAS GET BROWSE against deletion of the queue entry, deletion is delayed until the last SAS GET BROWSE task has ended. This is called **'delayed deletion'** and performed by the following steps:

- The queue entry is removed from its chain by the deleting task (IPW$DQS call).
- Then IPW$$FQ is called, which checks the MACC count and the 'delayed deletion' flag as follows:
  - If MACC > 0 and the queue entry was not yet flagged being in 'delayed deletion', the queue entry is not deleted, but flagged as being in 'delayed deletion' and the count of queue records and DBLK groups in 'delayed deletion' is increased. This count is shown in PDISPLAY STATUS report.
  - If MACC > 0 and the queue entry was already flagged being in 'delayed deletion', the queue entry is not deleted and the 'delayed deletion' counts are not updated.
  - If MACC = 0 and the queue entry was not yet flagged being in 'delayed deletion', the queue entry is deleted without updating the 'delayed deletion' counts.
  - If MACC = 0 and the queue entry was flagged being in 'delayed deletion', the queue entry is deleted and the count of queue records and DBLK groups in 'delayed deletion' is decreased.
- Module IPW$$LW calls IPW$$DQ to decrease MACC, each time a SAS GET BROWSE task issues its QUIT request for a queue entry in 'delayed deletion'. Afterwards MACC is checked and if zero, IPW$$FQ is called to free the queue entry.
- Queue file recovery takes care of queue entries in 'delayed deletion' by adjusting the number of SAS GET BROWSE tasks to zero for each system being recovered. The 'delayed deletion' counts are reset together with all 'delayed deletion' flags, so that calling IPW$$FQ for all queue entries in 'delayed deletion' rebuilds the correct counts and states.

Get In Creation Queue Entry: A special flavour of the direct SAS GET BROWSE request type is the direct SAS GET BROWSE for a queue entry 'in creation'. This request selects a queue entry, which is currently created by an execution writer task on the same system (if running shared). If the creating task TCB and the attached queue record fulfill certain conditions, the MACC and the queue entry attributes are checked against the specification of the requestor. If no such queue entry 'in creation' is found, the requestor is informed by the function return code byte of the TCB which is translated to a return code/feedback/feedback2 code passed to the SAS application program. If the queue entry is selected, the current spooling state is commited to disk in routine NQFRZ (part of IPW$$NQ).

1. The queue record copy of the creating task is copied to the queue record copy of the requesting task including fields QRCCNR, QRCCLC & QRCCPG (record count, line count, page count).
2. The MCB of the current spooled DBLK is locked by the requestor.
3. The DBLK in storage of the creating task is written to disk (using the DBLK area of the requestor of I/O Write) to make all data available on disk including the last record counted by QRCCNR, QRCCLC & QRCCPG.
4. The MCB is released.
5. The spooling state of the requestors queue record copy (QRNR, QRLC & QRPG) is updated using QRCCNR, QRCCLC & QRCCPG. This enables function Get Data Record, to present End Of Data, when QRCCNR or QRCCLC is reached.

Get Data Record Function (IPW$$GD): The function is invoked via the IPW$GDR macro instruction. The routine provides the calling routine with a logical record by means of the record request word of the TCB. If the DBLK area is exhausted a new physical record (DBLK) will be read from the data file. If the current DBLK group is exhausted, the next DBLK group is addressed via the SER.

If the logical record consists of more than one extension record, the original record is re-constructed before it is passed to the caller.

For each queue entry the current position is now counted in new fields QRCCNR, QRCCLC and QRCCPG. Module IPW$$LW observes these fields in its Restart Handler. If a queue entry 'in creation' is accessed, the current line/record number is compared with the number of lines/records to prevent reading past the last record committed on disk. When the last record/line has been passed to the caller, the next read requests will present End Of Data together with an artificial Job Trailer Record.

Time Event Scheduling

'Wait for Run' Subqueue: All queue entries of the VSE/POWER RDR queue are chained together via a forward and a backward pointer whereby for each class two class-chains exist, a dispatchable and a non-dispatchable class chain. In addition to these pointers exists a pointer which chains together all dispatchable jobs with a due date that has not yet expired. This chain is called the 'wait for run' subqueue and is built by forward pointers only, no backward pointers exist. The 'wait for run' subqueue contains queue entries of all classes, sorted only by the next due date, not by priority nor by class.

The 'wait for run' subqueue is maintained by the module IPW$$TQ, which is called by module IPW$$QM via IPW$ITQ whenever a queue entry is added or deleted to the local RDR queue.

The relationship and anchoring of the different queues and chains is laid out in the following Figure 37.

Note however that the 'wait for run' subqueue chains together queue entries of any class according to the next due date. In the figure below the pointers for the 'wait for run' subqueue are chosen in that way to keep the figure as simple as possible. Of course it may happen that the first element of the 'wait for run' subqueue is a queue record of class n, the second element a queue record of class m, the third element a queue record of class n, the forth element a queue record of class m, the fifth element a queue record of class o, the sixth element a queue record of class m, and so on.

If VSE/POWER terminates abnormally and full recovery is running when VSE/POWER is restarted, all queue entries are read in and all class chains are rebuilt. During this rebuilt the 'wait for run' subqueue is rebuilt automatically, but without calculating a new due date. After all chains are built, the TES task is attached and calls the INIT routine in IPW$$TQ which now examines the due date and schedules the jobs, if the due date has expired.
Jobs with an expired due date (and which were therefore not chained in the 'wait for run' subqueue) at the time VSE/POWER terminated abnormally, are not chained into the 'wait for run' subqueue at the time VSE/POWER restarts, but are chained in the dispatchable class chain for immediate processing. The same happens to jobs which have been in execution state.

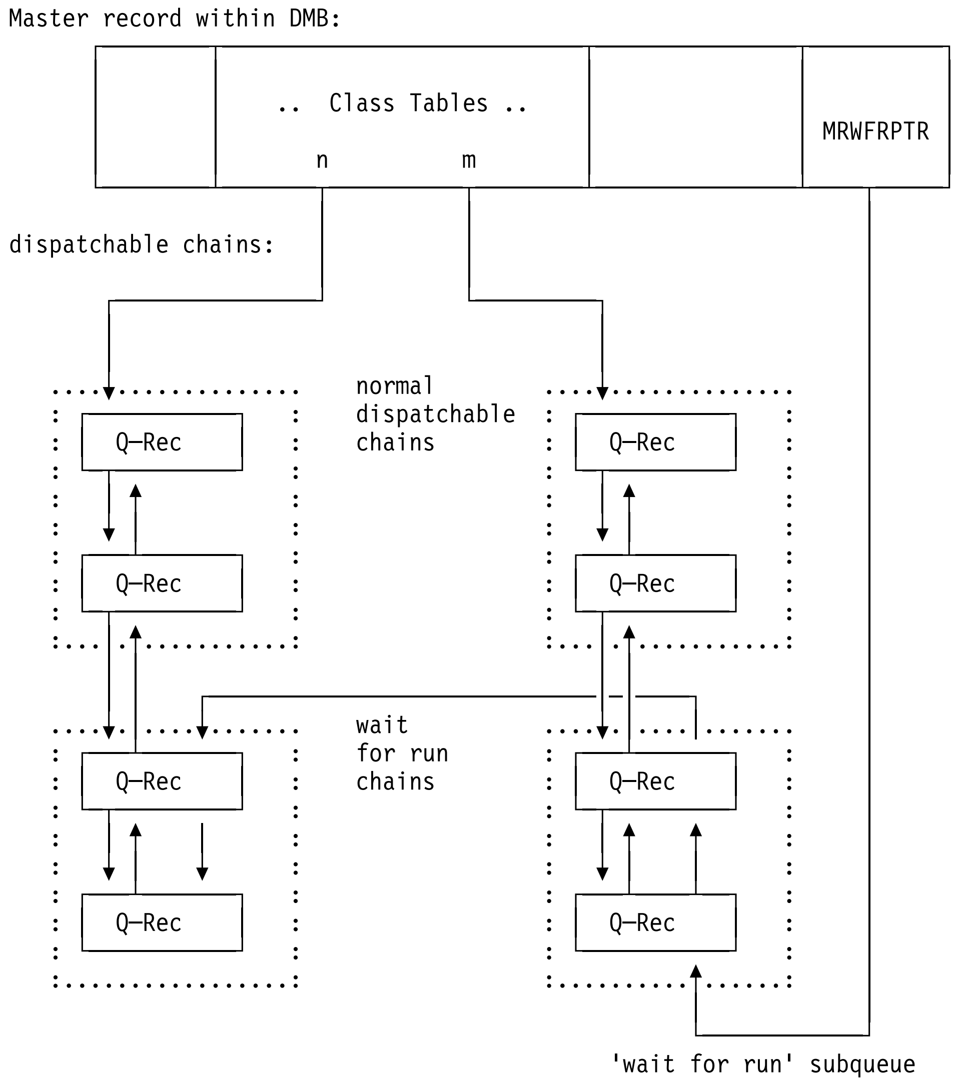


Figure 37. Structure of Job Queue and 'Wait For Run' Subqueue

TES Task: The time event scheduling (TES) task, is given control to make a job eligible for processing as soon as the due date expires. Therefore a time-interval is maintained for the first entry in the 'wait for run' subqueue. The TES task is attached during the initialization of VSE/POWER and detached during termination of VSE/POWER. When the TES task is attached it task calls IPW$$TQI to check if the due date of a queue entry in the 'wait for run' subqueue has expired. The TES task is attached even if the 'wait for run' subqueue is empty to avoid later on deadlocks which might occur due to storage constraints. The code for the TES task resides in module IPW$$TV.

Shared Considerations: To maintain the 'wait for run' subqueue in a shared environment fields and equates are used in the master record:

1. Whenever a system alters the first queue entry in the 'wait for run' subqueue, this system sets 'wait for run subqueue changed' for all other systems in the master record.
2. Whenever a system gets control in the T1-interval, the timer task (IPW$$TI) tests its 'wait for run subqueue changed' bit and if found, posts the TES task and resets its 'wait for run subqueue changed' bit.

Maintain 'Wait For Run' Subqueue Function (IPW$$TQ): This module maintains the 'wait for run' subqueue and is invoked by the interface macro IPW$ITQ. The 'wait for run' subqueue contains the dispatchable queue entries of the local RDR queue with not yet expired due date which are chained by forward pointers only. This means the 'wait for run' subqueue contains the queue entries of the 'wait for run' chains of all classes, but sorted according to the next due date, not sorted by class and priority. Whenever this module gets called, the DMB must be already locked. So far the following services are offered and used by IPW$$AQ, IPW$$DQ and IPW$$TV:

1. ADD: To complete the adding of a dispatchable queue entry with time event scheduling information to the local reader queue. If necessary, the queue entry is chained into the 'wait for run' subqueue:

   a. If recovery is active and the queue entry is not chained in the 'wait for run' subqueue, return with QRDG0W = OFF (not queued into 'wait for run' subqueue).
   b. If recovery is active and the queue entry is chained in the 'wait for run' subqueue:
      1) Chain queue entry into 'wait for run' subqueue according to the already existing next due date and time stamp of job creation.
      2) If first queue entry of 'wait for run' subqueue has been added:
         a) Set bit 'wait for run subqueue updated' for other shared systems.
         b) Post TES task to process new first 'wait for run' subqueue entry.
      3) Return with QRDG0W = ON (queued into 'wait for run' subqueue).
   c. If job runs more than once, calculate next due date.
      (Do not calculate next due date, if command processor is running and indication is not set that the calculation is necessary.)
   d. If just a due time is specified (i.e. no absolute due date and no cycling info specified), calculate next due date.
   e. If next due date not later than current date, return with QRDG0W = OFF (not queued into 'wait for run' subqueue).
   f. If next due date later than current date:
      1) Chain queue entry into 'wait for run' subqueue according to the next due date.
      2) If first queue entry of 'wait for run' subqueue has been added:
         a) Set bit 'wait for run subqueue updated' for other shared systems.
         b) Post TES task to process new first 'wait for run' subqueue entry.
      3) Return with QRDG0W = ON (queued into 'wait for run' subqueue).

2. DEL: To remove a queue entry from the 'wait for run' subqueue. The following steps have to be done:

   a. Remove queue entry from 'wait for run' subqueue.
   b. If first entry of 'wait for run' subqueue has been deleted:
      1) Set bit 'wait for run subqueue updated' for other shared systems.
      2) Post TES task to process new first 'wait for run' subqueue entry
         (if this request used by TES task, do not post the TES task).
   c. If the entry which should be deleted could not be found in the 'wait for run' subqueue, error message 1QZ0I with RC=18 is issued and VSE/POWER continues its normal processing.

3. INIT: To complete the initialization of VSE/POWER. This service is used only by the TES task and used just once, namely immediately after the TES task has been attached during the initialization of VSE/POWER. This routine is called in any case, no matter if VSE/POWER has made a warm start, partial or full recovery. This allows that during full recovery no new due dates are calculated, because this is done now at initialization of the TES task. In order to move those queue entries from the 'wait for run' chain whose next due date has expired, into the dispatchable class chain the following steps have to be done:

   a. Get current date and time using VSE/AF macro GETIME.
   b. Loop through 'wait for run' subqueue
      (The loop is left if 'wait for run' subqueue is empty or next due date is a date later than the current date and time):
      1) If RERUN=YES 'specified':

        a) Dequeue entry from 'wait for run' chain using macro IPW$DQS with option LOCATE.
        b) Add queue entry into normal dispatchable class chain using macro IPW$AQS (no need for IPW$$AQ (TCTI="YTES") to call IPW$$TQ once more, because VSE/POWER knows at this point already that the job has to be scheduled immediately).
    2) If RERUN=NO specified and the due date is today and the due time is before or equal to the current time:
        a) Dequeue queue entry from 'wait for run' chain using macro IPW$DQS with option LOCATE.
        b) Add queue entry to normal dispatchable class chain using macro IPW$AQS.
    3) If RERUN=NO specified and the due date was yesterday or earlier than yesterday:
        a) If no cycling information exists:
            i. Change disposition to H or L.
            ii. Dequeue entry from 'wait for run' chain using macro IPW$DQS with option LOCATE.
            iii. Add queue entry to non-dispatchable class chain using macro IPW$AQS.
        b) If cycling information exists:
            i. Do not dequeue entry from 'wait for run' chain.
            ii. Calculate next due date.
            iii. Chain queue entry into 'wait for run' subqueue anew.
            iv. Set refresh bits for other shared systems.

Note: No lock of the DMB is necessary because this routine is called at initialization time. Only the initialization task is attached at this time and the initialization task waits till it gets posted again by the TES task.

Time Event Scheduling Function (IPW$$TV): This module is the code for the TES task and maintains the time interval in order to move a queue entry from the 'wait for run' subqueue into the dispatchable class chain for immediate processing as soon as the due date has expired. The TES task is attached during the initialization of VSE/POWER and detached during the termination of VSE/POWER. The TES task is attached even if the 'wait for run' subqueue is empty to avoid later on deadlocks which might occur due to storage constraints.

The TES task calculates the time interval for the first entry of the 'wait for run' subqueue and waits till it expires. If it expires, the entry is removed from the 'wait for run' subqueue and chained into the dispatchable class chain for immediate processing.
Due to external events (PDELETE, PALTER or the execution processor of another shared system) the first entry of the 'wait for run' subqueue might have been deleted or changed. In this case the task ECB has been posted and a new interval is calculated. The task ECB is also posted during the termination of VSE/POWER when the task has to stop processing.

Note that only for the first job a time interval is calculated till its next expiration date. This interval might be a very large one. Due to the limited amount (ca. 7 hours) of the IPW$STM macro, an 'intermediate' interval might be calculated.

Following are more details for this module:

1. Call IPW$$TQI using 'IPW$ITQ INIT' to prime the 'wait for run' subqueue in all cases of cold-, warm-, partial or full recovery start.
2. Post initialization task for continuation.
3. Post own task ECB to enter ff loop once for first element.
4. Do forever (loop ONE)
   (loop left as soon as stop code exists)
    a. WAIT for task ECB to be posted
       (Task ECB posted if stop code or 1st element in 'wait for run' subqueue changed by $$TQ ADD/DEL actions or other shared system actions)
       (Task ECB also posted if time interval via TQE expires)

```
        1) Reset post bit of task ECB.
        2) If TQE-interval outstanding, cancel it.
        3) If stop code exists, leave loop (ONE)
        4) Lock DMB
        5) Do forever (loop TWO)
           (loop left if queue empty or job found with not yet expired due date)
           a) If 'wait for run' subqueue empty, leave loop (TWO).
           b) Compare due date of first entry in 'wait for run' subqueue with current date/time.
           c) If due date not yet expired:
                i. Set up new time interval, let task ECB be posted at expiration
                   (note: use minimum (VSE/AF-max, necessary interval for entry)!)
               ii. Leave loop (TWO)
           d) If due date already expired:
                i. Remove queue entry from its 'wait for run' chain using macro IPW$DQS with option
                   LOCATE (thereby entry removed from 'wait for run' subqueue by nested call 'IPW$ITQ
                   DEL')
               ii. Add queue entry into its really dispatchable class chain using macro IPW$AQS (call
                   from TES task, with conditional jobnumber increasing by $$AQ)
           e) Repeat loop TWO with next entry.
        6) Unlock DMB
    b. Continue to wait for next ECB post event (loop ONE)
5. Clean up and detach task (due to stop code 'S')
```

Notes:

1. Use the IPW$AQS and IPW$DQS macros with the option LOCK=NO to avoid the unlocking in IPW$$AQ and IPW$$DQ.

2. The lock of the DMB in loop TWO is necessary to maintain the integrity of the 'wait for run' subqueue during concurrent access of $$TQ ADD/DEL and of the TES task of the own system and/or of other sharing systems.

Running in ESA-Mode

Note: Only ESA mode is allowed.

Usage of ESA-Mode

VSE/POWER is running in ESA-mode when running in ESA supervisor mode on ESA/370 processors. VSE/POWER is not running in ESA-mode when running in ESA supervisor mode on XA/370 processors, because the XA/370 processors do not support access registers. The setting of the ESA-mode is done during the initialization of VSE/POWER (IPW$$I2).

Usage of Access Registers

When VSE/POWER is running in ESA-mode, the access registers are the only facility of the ESA-mode, VSE/POWER makes use of. VSE/POWER uses the access registers always regardless of running in the shared area or in private address space. The access registers are used in the following ways:

1. Modules running under the VSE/POWER task (e.g. execution tasks) use the access register to address data (e.g. CCB, CCW) in the partition controlled by VSE/POWER.
2. VSE/POWER appendages to supervisor services which run under the task of the user partition and not under the VSE/POWER task, use the access register to address data (e.g. main ECB of VSE/POWER, execution TCB) in the VSE/POWER partition.

Initially all access registers are set to zero. Only the following access registers are used:

1. Access register 7 always to address the CCB.
2. Access register 8 mainly to address the CCW.
3. Access register 1 a few times to address data in a partition.
4. Access register 6 a few times to address data in a partition.
5. Access register 5 to clear an access register to zero.

Access registers 1 and 6 are reset to zero as soon as they are no longer used to address data in the other partition.

Note that the usage of the access register is dependent on setting the access-register mode. The access-register mode is set on and off only by the execution tasks. The access registers are not cleared whenever a VSE/POWER task other than an execution task gets control (performance reasons). Therefore if such a task is running, the access registers might contain any value, but this value is never used for addressing, because the access-register mode is always set off.

A VSE/POWER subtask too, never uses any access register and does not set the access-register mode. Such a subtask should always have zeros in its access register (due to the initialization by the supervisor when attaching the task).

Whenever an access register is used in the code, the abbreviation ARx (x=0,1,..,F) is used (defined in macro IPW$EQU).

The access registers contain the ALET of the partition. The ALET is retrieved by using the VSE/SP Supervisor GETFLD macro.

If an abnormal termination of the VSE/POWER main task or a subtask occurs, the access registers are saved in the work area of module IPW$$AT.

Usage of Access Register in Modules

The following modules use the access registers in the following way:

1. IPW$$AT
   a. To save the access registers when an abend occurs.
   b. To address data in a partition controlled by VSE/POWER.
2. IPW$$TR
   To address data in a partition controlled by VSE/POWER.
3. IPW$$XJ
   To address data in a partition controlled by VSE/POWER.
4. IPW$$PD (part of IPW$$DM)
   To address data in a partition controlled by VSE/POWER.
5. IPW$$NU
   a. Dispatcher routines:
      1) To save (within the TCB) and restore the access registers.
      2) To restore the access-register mode.
      3) To store the access registers into the trace entry.
   b. Validation routines for CCW to address data in a partition controlled by VSE/POWER.
   c. Page Fault appendage to save the access registers and the access-register mode.
   d. I/O appendage (SVC 00 and 03) to address data in the VSE/POWER partition.
   e. Accounting appendage (SVC 90 and 91) to address data in the VSE/POWER partition.
   f. Hot Reader appendage to address data in the VSE/POWER partition.

   The above listed appendage routines within IPW$$NU use the access register only if VSE/POWER is running in a private address space. Other appendages than listed above (e.g. the BSC appendage) process supervisor calls which have been issued by the VSE/POWER task and all data addressed are located within the VSE/POWER partition. Therefore there is no need to use any access register in these appendages.
6. IPW$$XRE, IPW$$XWE
   The execution reader tasks and the execution writer tasks address data in the partition controlled by VSE/POWER, namely the CCB, the CCW and the data addressed by the CCW.
7. IPW$$MX
   To address an (incorrect) CCB or CCW in a partition controlled by VSE/POWER for issuing the message(s) 1R30I.

Addressing Exception in Access-Register Mode

The message 1Q2CI is issued, if VSE/POWER terminated abnormally. The PSW is displayed containing the address of the instruction which follows the failing instruction. This might not be true if running in ESA-mode: the address points to the failing instruction itself, because in some cases the hardware reports an error without having updated the instruction address within the PSW. This happens, if access-register mode is set on and an exception occurs during access-register translation (for example ASTE-sequence exception, detailed information see chapter 6 in *ESA/370 Principles of Operation*, SA22-7200). Such an exception occurs, if some of the tables maintained by the supervisor for access-register translation are corrupted. The various program-interruption codes returned by the hardware are usually translated by the VSE/SP supervisor into CC=20 (which means program check) and an 'addressing exception', displayed within message 1Q2CI. The original program-interruption code might be found within a debug entry of the supervisor.

Multiprocessor Support

External Invocation and Function

For getting acquainted to definitions as 'parallel work unit' etc., for how to activate and track MP Support, one should first read 'VSE/POWER Multiprocessor Support' as offered in *VSE/POWER Administration and Operation* manual.

Internal Implementation Overview

The following text is extracted from the header of module IPW$$XRE under 'Turbo Dispatcher Mode Usage Summary'. It reflects the support implemetentation 'as is', while a rationale for such chosen implementation is offered in subsequent 'Specification' and 'Design' chapters.

```
TURBO DISPATCHER SUPPORT IN VSE/POWER ...

     ... IS A BUNDLE OF FACILITIES THAT ALLOW PRIVATE
     SUBTASKS TO OPERATE (UNDER VSE/POWER MAINTASK) AS A
     PARALLEL WORK UNIT (PA) AS LONG AS POSSIBLE AND
     TO SWITCH TO A NON-PARALLEL WORK UNIT (NP) FOR SOME
     INSTRUCTIONS WHEN REQUIRED BY THE MULTI PROCESSOR (MP)
     ARCHITECTURE TO UPDATE
           - 1ST LOWCORE PAGE (SGLOWC, SYSCOM, BG-COMREG)
           - SUPERVISOR CONTROL BLOCKS
           - ISSUE IDUMP SVC
           - ENTER VTAM, LIBRARIAN, AND TRANSIENT SERVICES
     WITH THIS BEHAVIOUR VSE/POWER TRIES TO MINIMIZE THE
     TIME, WHERE THE SPOOLING SYSTEM RUNS AS NP WORK UNIT.


ATTRIBUTES OF TASKS IN THE VSE/POWER PARTITION

     *  MAINTASK      - AN ESA HYBRID TASK, BECAUSE IT HAS
                        PROTECTION KEY 0, BUT IS ENTITLED
                        TO SWITCH TO A PA AND NP WORK UNIT
     *  VSE SUBTASKS  - ATTACHED BY MAINTASK, PROPAGATING THE
                        PROTECTION KEY 0 TO THE SUBTASK, HENCE
                        SUBTASKS RUN AS NP WORK UNIT ONLY -
                        APART FROM THE TD-SUBTASK, WHICH SWITCHES
                        TO RUN PERMANENTLY AS PA WORK UNIT.
     *  PRIVATE TASKS - THEY GAIN CONTROL BY THE POWER TASK
                        DISPATCHER, THEN TASKS OPERATE AS
                        THE VSE/POWER MAINTASK
```

```
INTERFACES TO SUPERVISOR TASKS & SPOOLED PARTITION TASKS

     READ THE DETAILED CODE COMMENTS FOR THE FF INTERFACES
       - AB|OC|TI-EXIT HEADERS OF THE POWER MAINTASK IN
                       MODULE IPW$$AT.AT250, AND
                       MODULE IPW$$I2.TIME4, AND
                       MODULE IPW$$NU.TI00 OR .STXOC000
       - VARIOUS SUPERVISOR APPENDAGES IN IPW$$NU (!)
       - JOB CONTROL EXIT AND ATTENTION RTN I/F IN IPW$$NU (!)


ACTIVATION OF TURBO DISPATCHER (TD) SUPPORT IN VSE/POWER

     THE DESCRIBED CODE SUPPORTING MULTI PROCESSORS
       - IS DRIVEN, WHENEVER 'TURBO DISPATCHER' HAS BEEN
         ACTIVATED DURING IPL AND POWER MP SUPPORT HAS
         BEEN ENABLED BY 'SET WORKUNIT=PA', AS RECORDED
         BY PDISPLAY STATUS:
            'PRESENT SESSION START (TURBO-DISP.-PA) ON ...
       - BYPASSES 'TDSERV SWITCHNP/PU' REQUESTS, WHENEVER
         TURBO DISPATCHER HAS NOT BEEN ACTIVATED AT IPL
       - BYPASSES 'TDSERV SWITCHNP/PU' REQUESTS, WHEN TD
         ACTIVATED BUT DEFAULT 'NP'-ONLY STARTUP DONE


WORK UNIT RULES FOR POWER PRIVATE TASKS

     * INIT/TERM TASK - ACQUIRES PROT. KEY 0 AT $$I1 TIME,
                        CONTINUES THEN AS NP WORK UNIT,
                        AND NEVER GIVES UP NP MODE AGAIN
     * GENERAL TASK   - ATTACHED AS PARALLEL WORK UNIT,
                        SWITCHES BY 'IPW$TDM NP' SERVICE-
                        CALL TO NP MODE ONLY THEN, WHEN
                        PROCESSED CODE REQUIRES - RETURNS
                        TO PA MODE BY 'IPW$TDM PU' CALL
     * EXCEPTION TASKS- ALL RJE/SNA TASKS (SEE $$SN) AND
                        PNET SNA CONN/DISCONN TASKS, THAT
                        OPERATE NON PARALLEL EXCLUSIVELY


SWITCH WORK UNIT BY 'TDSERV' MACRO - CODED IN MODULE ...

     1) IPW$$NU MODE SWITCH SERVICE, ENTERED BY MACRO CALL
                       'IPW$TDM NP/PU' (R0,R1,R2 DESTR'D)
     2) IPW$$ID LOCAL 'TDSERV SWITCHNP/PU' SVC TO AVOID
                        RECURSIVE SERVICE ENTRY TO IPW$$NU
     3) IPW$$AT LOCAL 'TDSERV SWITCHNP' SVC, WHICH CANNOT
                        RELY ON ANY CODE OUTSIDE IPW$$AT


FUNCTION OF 'MODE SWITCH SERVICE' IN IPW$$NU

     - DEPENDING ON IPW$TDM NP/PU REQUEST, TO CHANGE WORK
       UNIT TYPE BY TDSERV SWITCHNP/PU SVC (BYPASS SVC,
       IF LOWCORE SHOWS 'DESIRED MODE ALREADY ACTIVE')
     - RECORD ACQUIRED WORK UNIT TYPE IN TCB BY ...
               ... TCF16.TCF16NP = 1, MEANING NON-PARALLEL
               ... TCF16.TCF16NP = 0, MEANING PARALLEL
```

```
LIFE SPAN OF A GENERAL POWER TASK ...

     - ENTERED INTO TASK DISPATCH CHAIN WITH TCF16NP=0=PA
       BY IPW$$NU TASK INITIATION ($ATT) SERVICE
     - $$NU.TM90 TASK DISPATCHER REQUESTS 'IPW$TDM PA' TO
       GIVE CONTROL TO TASK AS DESIRED WORK UNIT
     - TASK PROCESSING CODE REQUESTS 'IPW$TDM NP' WHEN
       MP ARCHITECTURE REQUIRES AND RETURNS TO PA AGAIN
     - WHEN TASK LOSES CONTROL BY PAGE FAULT OR ANY $WFX,
       ITS RECORDED TCF16 WORK UNIT TYPE IS RE-ESTABLISHED
       WHEN $$NU.TM90 DISPATCHER SELECTS TASK AGAIN


LIFE SPAN OF AN 'NP' EXCEPTION TASK

     - EXCEPTION TASKS (INIT/TERM, ALL RJE/SNA, PNET SNA
       CONN/DISCONN TASK) ARE COLLECTED IN 'NP-MUST' LIST
     - NP-MUST TASKS ARE ENTERED INTO TASK CHAIN BY $ATT
       WITH TCF16NP=1=NP
     - WHENEVER PROCESSING CODE DRIVEN BY SUCH TASKS
       CALLS 'IPW$TDM PU', THEN MODE SWITCH SERVICE WILL
       IGNORE SWITCH TO PARALLEL FOR NP-MUST TASK


MP RULES WHEN CALLING SERVICES OR FUNCTIONS

     - IPW$$NU SERVICES DO NOT SWITCH MP MODE AT ALL,
       IT IS RECOMMENDED TO CALL THEM AS PA WORK UNIT
     - POWER FUNCTIONS EXPECT TO BE CALLED AS PA WORK UNIT,
       THEY SWITCH TO NP WHEN REQUIRED AND RETURN PA
```

Internal Functional Specifications

The following extract of the Multiprocessor Support Specifications shows, which rules any VSE/POWER code has to obey to when running generally in parallel mode, but interfaces with other services and Supervisor appendages for VSE/POWER.

With respect to VSE/POWER the following holds for Turbo Dispatcher processing:

- During IPL either the 'Standard' or 'Turbo' Dispatcher (TD) is selectable
- TD means no change to VSE/POWER residing in either shared or private partition
- TD dispatches an entire partition to a single CPU, that means, VSE-maintask and VSE-subtasks of same partition do not gain control in parallel
- Because the VSE/POWER Maintask acquires protection key zero (as ACF/VTAM), it is dispatched as Non-Parallel (NP) Work Unit, but is entitled to request Parallel Mode (get dispatched as Parallel Work Unit).

Code of Private Subtasks (Power Task): The Parallel Support (also called MP Support) of VSE/POWER must be enabled with the autostart statement

```
SET WORKUNIT=PA
```

Then the VSE/POWER partition maintask acquires protection key 0 during VSE/POWER initialization and therefore enters NP state processing automatically. When starting VSE/POWER's private subtasks under control of the maintask, these subtasks (also named 'tasks') call the VSE/POWER service IPW$TDM PU to drive the SVC

```
TDSERV FUNC=SWITCHPU
```

in order to enter their function code in parallel mode. Now the VSE/POWER maintask is a hybrid system component - using protection key 0, but yet running in parallel mode. Code areas requiring non parallel processing are identified according to the needs listed below. When a task enters and leaves an NP code area, mode switching is requested by IPW$TDM NP/PU (resulting in corresponding TDSERV SVC) and its mode state is recorded in the task control block (TCB). When a task loses control by a page fault or any VSE/POWER wait condition, its recorded mode is re-established, when the task is given control back by the VSE/POWER task dispatcher.

Code processed by a VSE/POWER task has to watch PU or NP processing according to the following rules, which are imposed by the VSE/ESA Supervisor and Turbo Dispatcher:

1. SVC calls decide on their own, when PU or NP is needed, and the original mode is re-installed upon return; exceptions SVC's are and must be called as NP work unit:
   a. SVC 2   .. call IDUMP, Fetch B-Transient
   b. SVC 43 .. DYNCLASS ID=GET, where Librarian calls are involved
   c. (SVC 107 .. for SENTER/SLEAVE only, not used by POWER)
2. BALR interfaces as entry to components that rely on operating in NP mode due to using protection key 0 (as VTAM, Librarian, or Transients, which must be entered in NP mode
3. VSE/POWER coded Supervisor exits as OC-, IT-, or AB-exit are entered with the currently active mode of the VSE/POWER maintask (or active mode of the VSE Subtask in case of IT-exit). Exits should be left for task continuation in the same mode
4. Prefix page (SGLOWC, SYSCOM, and BG-COMREG) changes (not reference!) must be done by NP code to let updates become effective on other CPUs
5. Updates in other Supervisor control blocks should also be done by NP code, and not by Compare & Swap (CS) like instructions, because counterpart code has not been implemented yet
6. Whenever parallel VSE/POWER work units overlap with NP VSE/POWER appendages, prefer CS-like instructions versus NP switch, in order to achieve CPU serialization for critical update code
7. VSE/POWER Page Fault Handling Overlap (PHO) appendage is entered as NP unit for pre-processing under control of POWER maintask
8. VSE/POWER Page Fault Handling Overlap (PHO) appendage is entered as NP unit for post-processing under Page Mgr. Task control, hence may overlap with parallel POWER code.
9. Asynchronously entered routines generally operate in NP mode, and may hence overlap with parallel POWER code. This is especially true for the VSE/POWER appendages in IPW$$NU, because they were called by an SVC:
   a. Job Control exit entered NP under spooled partition control
   b. Attention I/F Routine entered NP under control of Att. rtn.
   c. BSC/CTC channel end appendage entered NP under control of I/O Supervisor
   d. Hot reader appendage entered NP under control of I/O Supervisor
   e. SVC 0/3 appendage entered NP under control of spooled partition
   f. SVC 90/91 appendage entered NP under control of spooled partition

Code of VSE Subtasks: VSE/POWER tasks attach VSE Subtasks for various internal reasons:

1. VTAM ACB Open/Close and Exit processing of PNET or RJE/SNA
2. Librarian, Dump, and Timer services
3. A bunch of so-called 'asynchronous' services
4. To provide the TCP/IP interface for the PNET TCP function
5. To provide the TCP/IP interface for the PNET SSL function

Actually attached by the partition maintask running with protection key 0, the 1) - 3) VSE Subtasks are given control with the same key and hence in NP mode. VSE/POWER code driven by VSE Subtasks is not heavily used. Therefore, and for reduced complexity, **no** switching to parallel mode is introduced in this code. However subtasks 4) and 5), the TD-Subtask and SD-Subtask are heavily used and do not update lowcore and Supervisor areas. To increase the total system's PA/NP share, these subtasks switch back to PA mode.

Apart from the IPW$$NU exit and appendage rules, all previously given mode rules have to be respected also in VSE Subtask code. Especially the following is true:

1. Due to Turbo Dispatcher dispatching on partition basis, VSE Subtask and VSE/POWER Maintask code always runs serialized, and may be interrupted at any instruction. Then PU maintask code may be dispatched, even if a higher priority NP VSE Subtask is ready to run, but waits for NP processing.
2. VSE/POWER's VTAM exit code is given control under the ACB Open/Close VSE Subtask and hence processes as NP unit

Code Common to Private and VSE Subtasks: Even such code exists in VSE/POWER for calling of VTAM SEND/RECEIVE requests. When given control under a VSE Subtask, the code is processed as NP unit. Upon entry and exit by a private subtask switching to NP and finally back to parallel mode (PU) is done.

Code of User Exits or OEM Hook Code: Code of VSE/POWER user exits or VSE/POWER Vendor hook code usually runs as VSE/POWER private subtask, for which PHO-processing and VSE/POWER task dispatching takes place. These functions keep track of the parallel or non-parallel mode at subtask interruption and re-dispatch time. This is only feasible, if switching of work units is not requested by the VSE TDSERV FUNC=SWITCHNP/PU macro, but by the corresponding VSE/POWER service macro IPW$TDM.

Internal Implemented Design

The following extract of the Multiprocessor Support design material discusses the processing rules imposed by the Specifications and justifies the chosen and implemented design. This design however may well be challenged for modification and extension one day!

Principles of Mode Switching for POWER Private Subtasks (Power Tasks)

General Overview: POWER tasks are generally attached in parallel mode for the code they should start with as a PU work unit. At entry to a NP work unit mode switching is requested by IPW$TDM NP, which is recorded by TCB flag TCF16NP=ON. At exit from the NP work unit mode switching is requested correspondingly by IPW$TDM PU, which is recorded by TCF16NP=OFF.
At page fault in NP (or PU) work unit, the IPW$$NU Page Fault pre-processor (under control of failing POWER maintask, in NP mode) puts task 'P'-bound asleep and leaves TCF16NP untouched. The POWER maintask continues in previous mode in POWER dispatcher code and selects next task to run. At 'TM90' task dispatcher decides by TCF16NP of next task, what work unit should be continued and requests corresponding mode switching by IPW$TDM. Using look-aside to Lowcore, the current NP/PU mode is checked, and superfluous TDSERV FUNC=SWITCHNP/PU may be bypassed for performance reasons.
Any call of IPW$WFx wait service also puts task asleep with current mode recorded in TCF16NP, and task dispatch at 'TM90' re-establishes the desired NP/PU mode as discussed before.

When any other VSE/POWER service or function is called, it should be entered as parallel work unit (PU). The called code will decide locally, when NP mode has to be entered and left again, and will always return as PU unit to the caller.
In principle VSE/POWER should run as parallel work unit for heavily used code wherever possible.

Initiator/Terminator Task: This task drives modules $$I1-$$I7, IP, and finally $$T1. Various prefix page and Supvr. Ctl. Block updates are done in performance uncritical code. To reduce complexity, this task should permanently run in NP mode! Follow up how TAIT task rises:

1. VSE/POWER starts under part. maintask in POWER Loader phase as normal PU partition
2. maintask acquires prot. key 0 near IPW$$I1-'I130' and continues NP (by Supvr. decision)
3. builds initial TATM-TAOC-TAIT TCB-chain near IPW$$I2-'I230'
   a. with TCF16NP=OFF for TAOC-TCB
   b. and requests IPW$TDM NP to re-enter NP mode (dummy request) and set TCF16NP=ON for own TAIT task
   c. and enforce default 'non-parallel-processing' for all POWER (as if SET WORKUNIT=NP been given later in $$I2) by setting CAT CAF4WKNP=ON
4. continues as POWER task by first IPW$WFD near IPW$$I2-'I238A'
5. may then hit a SET WORKUNIT=PA autostart statement that sets CAT CAF4WKNP=OFF!
6. for never loosing NP mode while calling any POWER service/function, extra code of $$NU-TDM service ignores an IPW$TDM PU request, if caller is TAIT-task
7. all $$I7 born tasks are initiated by the IPW$ATT macro, which ensures TCF16NP=OFF
8. PHO is enabled not before end of $$I7

IPW$$NU-'PN10' Service: The IPW$TDM macro call enters the IPW$$NU 'NP/PU Mode Switching' (also TDM-) Service via the CAT service branch table entry 'PN00' and acts as follows:

```
1) If Turbo Dispatcher not activated, ignore request.
   If all session non-parallel, that means CAF4WKNP=ON,
   then ignore request.
2) If Init task requests 'PU' mode, ignore request.
3) If certain PNET SNA or all RJE/SNA tasks (must always
   run 'NP'!) request 'PU' mode, then ignore request.
4) If desired mode is already active, bypass TDSERV.
5) Request TDSERV FUNC= according to passed register 1 values.
6) Set TCF16NP according to desired and acquired mode.
```

Dispatcher Mode Switch: Irrespective of NP/PU mode the dispatcher code has operated in, while selecting the next task, 'TM90' task selection will always call IPW$TDM service to obtain that mode for the POWER maintask, which the task being dispatched has recorded by TCF16NP as 'mode to continue with'.

IPW$ATT Task Initiation: Certain PNET SNA and all RJE/SNA tasks (see VTAM BALR interface) must always run in NP mode. IPW$$NU task initiation attaches all tasks of this 'NP-MUST' list with TCF16NP=ON, what takes effect at Task Dispatch Time when task enters its function code. When this task enters any Service/Function code that calls IPW$TDM PU, then $$NU Mode Switching Service ignores such request.

NP Mode for Prefix Page & Supvr. Ctl. Blk. Update by POWER Task

Module List for SGLOWC Update: This first part of prefix page is only REFERENCED (not updated) by module IPW$$NU,-LW,-CS,-CPS. Hence no change is required!

Module List for SYSCOM & COMREG Update: Modules and code areas driven by POWER Tasks requesting such update are mentioned with 'Supvr. Ctl. Blk. Update'.

Module List for Supvr. Ctl. Blk. Update: The following modules driven by POWER Tasks are found to do such update. Insert IPW$TDM NP before - and IPW$TDM PU after update. For performance reasons avoid frequent switching!

1. IPW$$CE - 1 occurrance
2. IPW$$TR - 5 occurrances
3. IPW$$XJ - 11 occurrances
4. IPW$$DP - 3 occurrances
5. IPW$$CS - 2 occurrances

6. IPW$$XRE - 18 occurrances

NP Mode for BALR Interfaces of POWER Task

Module List BALR I/F: Branch And Link to code running NP exclusively from the POWER maintask (with key 0 and yet running PU) confuses the called code. So we request IPW$TDM NP, or run NP exclusively, when entering such BALR interface for:

- PNET SNA VTAM BALR interfaces: Fortunately Receiver and Transmitter code does not request VTAM macros, but queues/de-queues buffers which are sent/received by the Line Driver Task in IPW$$SR. So IPW$$NR and IPW$$NT code may run always PU. Line Manager task may process PU in IPW$$LD and in $$LD1-$$LD5 but switch locally to NP mode in $$SR.
  $$S2 and $$S3 Session Connect/Disconnect Tasks request VTAM macros. These tasks are performance uncritical and run always NP, which is guaranteed by $$NU 'Task Init' and $$NU 'Mode Switch' that respect a list of tasks, for which NP is a MUST. So we summarize:
  1. IPW$$S1 - 'ACB-Open-Close' always NP because VSE/Subtask
  2. IPW$$SE - exits driven NP by VSE/Subtask
  3. IPW$$SR - see extra discussion as 'Common Code Module'
  4. IPW$$S2 - 'Connect Task' shall always run NP --> 'NP-MUST'
  5. IPW$$S3 - 'Discon. Task' shall always run NP --> 'NP-MUST'
- RJE/SNA VTAM BALR interfaces: Unfortunately the SNA Manager ($$SN) and all its attached POWER tasks (see IPW$$SN-'LGH'), namely ....
  1. 1LGH - IPW$$LH logon processor 1
  2. 1LGN - IPW$$LN logon processor 2
  3. 1LGF - IPW$$LF logoff processor
  4. 1MSG - IPW$$MP message processor
  5. nRDR - IPW$$IB inbound processor
  6. nLST - IPW$$OB outbound list processor
  7. nPUN - IPW$$OB outbound punch processor

  ... issue VTAM macro requests spread all over the modules. Running PU and switching locally NP makes little sense, because of the small 256-byte RU-size exchanged with VTAM. Therefore all mentioned tasks will run NP exclusively by entering them into the 'NP-MUST' list.
  For the RJE/SNA VSE/Subtask in code part two of $$SN respect:
  1. IPW$$SN - 'ACB-Open-Close' always NP because VSE/Subtask
  2. IPW$$VE - exits driven NP by VSE/Subtask
- Librarian/ICCF BALR interfaces:
  1. $$AS - actual SLI BALR interfaces are driven by VSE ICCF/Librarian subtask, which always runs NP
- IPW$$LU calling non-POWER service code
  1. 'LU132' - enter IJBSSYS mode PHASE after IPW$TDM NP request

NP Mode for Exception SVC's called by POWER Task

Module List Except. SVC: Some SVC's expect to be called by key 0 task with NP mode. VSE/POWER takes mode provision when calling:

- SVC 2 - call IDUMP/BAM Transient in ...
  1. IPW$$AS - Dump Subtask - runs NP because VSE Subtask
  2. IPW$$ID - driven by VSE Subtask, then always NP
  3. IPW$$ID - driven by $$AT call for VSE Subtask, always NP
  4. IPW$$ID - driven by $$AT call for POWER Task, NP switch already done in $$AT
  5. IPW$$ID - driven by POWER task with extra switch required in IPW$$ID

- DYNCLASS ID=GET in ...
    1. IPW$$AS - driven as async. service request by NP subtask

Principles of Processing for VSE-Subtasks

General Overview: Attached by VSE/POWER maintask with key zero, Supervisor passes control to VSE-subtasks with same key and as NP work unit, which is never changed by VSE/POWER code request. The following modules are driven by the corresponding VSE-subtask and need therefore not be checked and changed at all:

1. Asynchr. Service Subtask
    a. IPW$$AS, label 'SUBTASK'-'IDUSUTA',
    where also DYNCLASS ID=GET (exception) SVC is requested
2. Dump Subtask
    a. IPW$$AS, label 'IDUSUTA'-'LBSSUB'
3. ICCF/Librarian Subtask
    a. IPW$$AS, label 'LBSSUB' till module end
4. Timer Subtask
    a. IPW$$TI, part II
5. PNET SNA SUBTASK
    a. IPW$$S1 (ACB Open, enable VTAM exits, ACB Close)
    b. IPW$$SE (VTAM exits code)
    c. IPW$$SR, Send-RPL-exit + Receive-RPL-exit
6. RJE/SNA SUBTASK
    a. IPW$$SN, part II (ACB Open, enable VTAM exits, ACB Close)
    b. IPW$$VE (VTAM exits code)
7. PNET TCP (TD-) SUBTASK switching to PA mode
    a. IPW$$TD
    b. IPW$$TS
    c. PNET SSL (SD-) SUBTASK switching to PA mode
        1) IPW$$SD
        2) IPW$$SS

Principles of Processing Common (VSE- and POWER subtask) Code

IPW$$AT: Actually driven as AB-exit or branched to by POWER task using IPW$CNC, this module is entered

1. NP/PU by failing VSE-Subtask ---> to be cancelled
2. NP/PU by failing POWER task ---> to be cancelled
3. NP/PU by failing POWER task in user exit ---> to resurrect

IPW$$AT changes Supvr. Ctl. Blocks, but is performance uncritical, therefore

- at entry by maintask use TDSERV FUNC=SWITCHNP, set TCF16NP
- leaving IPW$$AT for exit recovery, let caller of user-exit switch back to PU mode

IPW$$ID: Module entered for IDUMP (SVC 2, needing NP, if POWER maintask) by

1. IPW$IDM request of VSE-SUBTASK(s) in NP mode, apart from TD-Subtask and SD-Subtask in PA mode
2. call from IPW$$AT (VSE-subtask or POWER task) in NP mode
3. IPW$IDM of POWER task in PU mode (as defined for functions), which needs switch to NP by local TDSERV

Then common code requests IDUMP SVC 2, and module is left again by three exit paths, where in case of call by POWER task we switch back PU by TDSERV - unless task is in NP-Must list.

IPW$$SR: Module contains interlocked code to drive the VTAM Send and Receive function - each called either by POWER LD-task or VTAM exit under ACB Open Subtask. Due to BALR I/F for SEND/RECEIVE macro, NP mode is required, which exists for VSE-Subtask call, but must be invoked for LD-task call:

1. at IPW$$SR-'SEND', successful LD-task entry use IPW$TDM NP
2. at IPW$$SR-'SEEXIT' (common!) exit, split up exit acc. to caller, and acquire PU mode by IPW$TDM for LD-task
3. at IPW$$SR-'RECV', successful LD-task entry use IPW$TDM NP
4. at IPW$$SR-'RECVEX' (common!) exit, split up exit acc. to caller, and acquire PU mode by IPW$TDM for LD-task

IPW$$MX Message Modification Part: From IPW$$MX-'MXENTRY' till end-of-module, code may be used by POWER task and VSE-Subtask (PNET|RJE/SNA). This code never requires NP mode. Therefore NO change is required for POWER task entering this module in any mode, or VSE-subtask entering this module in NP mode, or even PA mode if TD-Subtask for PNET TCP or SD-Subtask for PNET SSL.

Principles of Processing for STXIT Exits: These exits are entered under control of the task that has established the exit by STXIT xx macro. Entry may be NP/PU, depending on the work unit the task currently processes.

- when exit code does not change processing mode, EXIT xx will pass control to the point of interrupt in original mode
- when exit code changes processing mode, it should re-establish the entry mode before EXIT

Following exits can be found in POWER:

1. AB-EXIT - for leaving exit and handling of common code see 'Common Code'-IPW$$AT
2. IT-EXIT-'CAIT' - setup by maintask in $$I2, used for posting of SETIME event in $$NU-'TI00'. No mode switch required in this code.
3. IT-EXIT-'TIME4' - setup temporarily in $$I2 for Q-file-LOCK. Is always called in NP mode, because driven by initial TAIT-task, which never loses NP mode.
4. IT-EXIT-'TIME4' - setup in $$TI by Timer-Subtask for Q-file-LOCK always called in NP mode. NO change required.
5. OC-EXIT-'CASTX00' - setup in $$I2 for MSG Fx,DATA=.... to submit diagnosis display commands. Entered NP/PU mode under any POWER Task:
   a. picks up predefined TCB 'STCB', not chained in TCB chain
   b. leaves old TCB still in 'R' state, unless OC-interrupt occurred during processing in $$NU task dispatcher
   c. enters $$CM for cmd parsing
   d. issues console I/O by DOS Wait (no chained TCB!)
   e. PHO not handled for 'STCB' TCB
   f. enters $$CD for desired display
   g. returns to $$CM and back into actual OC-Exit for EXIT OC

   All the processed OC-exit, $$CM, and $$CD code does not change the processing mode valid at OC-exit entry. No change is required.

Principles of Processing for SUPVR Appendages in IPW$$NU

Such code is usually not driven by the POWER maintask but by Supervisor tasks or user partition tasks.
On uni-processor this code runs pseudo-parallel to VSE/POWER, that means, it may interrupt the POWER maintask at any point in time (after instruction fully completed). Because of interrupts disabled and no page fault happening, the appendage code will complete totally before giving control back to the POWER maintask.
On multi-processor this code operates NP with interrupts disabled and no page fault happening, and if POWER counterpart code ...

- runs PU (in parallel on other processor), then both code parts may overtake each other (depending on processors speed). Even the instruction boundary is not guaranteed - unless 'CS' and 'TS' instructions are used.
- runs NP, then POWER code may be interrupted at any point in time (after instruction FULLY completed). Again appendage code will complete TOTALLY before giving control back to POWER
- runs NP with 'interrupts disabled' (no page faults allowed then - POWER code to be called by SVC or driven like appendage code), then POWER code will not lose control at all! Note, such code exists only as few Supervisor Appendages of VSE/POWER, that run under POWER maintask.

PHO-Preprocessor (see $$NU-'PF01'): Entered NP under POWER maintask, therefore no parallelism at all, and no code change.

PHO-Postprocessor (see $$NU-'PF03'): Entered NP under Page Mgr. Task. Sets page-fault-solved task 'D' bound, follows TCB chain for another 'P' bound TCB. At same time the POWER maintask may on another processor modify the TCB chain by

- $$NU-'TA01' Task Init. ($ATT), which always guarantees a correct TCB forward chain
 --> no code change
- $$NU-'TD01' Task Term. ($DET), which removes e.g. TCB-x from chain and soon clears the TCB-x storage. PHO appendage cannot trust the next TCB pointer in TCB-x, if POWER ran PU
 --> in Task Term. at 'TD01A' call IPW$TDM NP to modify TCB chain as NP work unit

End-of-JOB Exit (see $$NU-'EOJ00'): Entered by Job Control during EOJ processing as NP unit under spooled partition maintask as re-enterant code to update the POWER Part. Ctl. Block of this partition. Any PU POWER IPW$$XRE code on other processor uses ECB wait/post logic to communicate with Job Control of the spooled partition --> no code change

Attention I/F (see $$NU-'AI00'): Entered NP by ATTN Rtn. Task at $$NU-'AI00' for non re-entrant processing with interrupts disabled and no page fault, since all AR buffers and referenced POWER areas are pfixed.
On uniprocessor, counterpart POWER cmd. processor can be interrupted after any completed instruction for giving control to the ATTN I/F code, which then completes ALL its processing before cmd. processor can continue.
On multiprocessors, while PU IPW$$CM on e.g. CPU1 is between any instruction or even instr. cycles, the NP ATTN I/F code may run on CPU2 either partly or totally. Hence code at 'AI30-AI42' may get in trouble with its counterpart in IPW$$CM at 'CPR970', where synchronization is established via flags of CPFG2
--> set IPW$$CM-'CPR970' NI instructions into NP mode and switch back by IPW$TDM PU. This is no performance loss, because a central operator command appears seldom.

BSC/CTC Channel End Appendage (see $$NU-'CE00'): Entered at $$NU-'CE00' as NP unit under I/O Supervisor task as NON re-enterant code, interrupts disabled. Task handles I/O events serialized, but CCB Post-bit and CSW-status or Sense-info are only set in CCB, 'after' control has been returned from appendage to Supervisor.

- For RJE appendage, posts TALM-TCEB, adds I/O partly completed LCB to end of TALM-CHEND queue, posts POWER partition by TREADY
  --> change code to post TALM-TCEB only after LCB chained ('CE70'), to cope with the PU work unit of IPW$$LM-'LM15'
  --> change code in Line Manger, which dequeues the oldest LCB from top(!) of TCBQ channel-end-chain, what cannot be gated against appendage by CS (as done for PNET). Therefore set IPW$$LM-'CHEND' de-queueing sequence into IPW$TDM NP mode and back to PU again. Thereby the CCB traffic bit is already posted, before the CCB is getting interpreted.
- For PNET BSC or CTCA - where input buffer (belonging to partly posted CCB of NCB) is queued on top of the PNET-Driver channel-end-queue (using CS already), where PNET Driver is posted, and waiting Power partition dispatched by TREADY
  --> no change required, since PU work unit of Line Driver IPW$$LD dequeues buffers with CS in 'LDBUFPR' routine
  --> change required not to interpret CCB by Line Driver before final posting of Traffic bit. Therefore check in BSC/CTC-IPW$$LD1 for NCB-CCB traffic bit already posted. If not, then enter IPW$WFE for CCB-post-bit.

Hot Reader Appendage (see $$NU-'HR00'): Entered NP by serialized I/O Supvr Task for non reenterant processing, whenever a physical reader device presents 'device end', while no READ request outstanding. That means, more cards have been put into the physical reader. The appendage communicates with IPW$$PR-'PR64' (having requested 1Q34I/1Q35A).
The following interface changes are introduced to resolve existing and new multiprocessor problems:

1. --> remove TCDVE from TCG2. Instead use TCDVEB communication byte to get rid of OI/NI instructions
2. --> exchange $WFI waiting in $$PR by $WFE TCEB waiting, and post Task ECB unconditionally from Reader Appendage
3. --> run TCDVEB & $WFE sequence in $$PR ('PR70'-'PR85') as NP work unit, meaning 'serialized' with NP Hot RDR Appendage. This is no performance impact. 1Q34I waiting event appears seldom.

SCV 0/3 Appendage (see $$NU-'SU00'): Entered NP by spooled partition task at $$NU-'SU00' with interrupts disabled, no page fault happening.

For SVC 0 request, the POWER counterpart code in IPW$$XRE/XWE clears TLCB when current spooled I/O CCB is handled, and only then accepts a new I/O request to be passed by the appendage. So there is no real need for POWER code to run non parallel. However for safety and respecting, that subsequent TREADY request enters NP mode anyhow ...
--> switch to NP at $$XRE-'XQ72' for CCB posting, TLCB clearing, and TREADY of spooled partition
--> switch to NP at $$XWE-'XX83' for the same reasons.

For SVC 3 - Quiesce request by Job Control, appendage code accepts it only, when counterpart exec. reader has a read I/O pending, cannot select a next job (1Q34I), and has flagged itself POWWPART (waiting for work) in spooled partition Comreg FLG1. Comreg update is done in NP mode anyhow, hence
--> no additional provisions required. Appendage will post task ECB with X'20' and $$NU Q-state processing will look aside and dispatch the exec. reader from $$XRE-'XQ16'.

SCV 90/91 Appendage (see $$NU-'SU90'): Entered NP under spooled part. task with interrupts disabled and no page fault expected - only requested by Job Control for

1. SVC 90 from PUTACCT macro (expected to come from $JOBACCT rtn. only) with R0->ECB|LEN|A(PUTACCT-Info) and R0 high order byte flagged X'90'
2. SVC 91 from Job Control (at end-of-job-step) with R0->ECB and R0 high order byte flagged X'91'

Appendage code passes request unconditionally to exec. reader TLCB. Explanation is, that these SVC's may only arrive from Job Control when no spooled read I/O is outstanding, with IPW$XRE to be in $WFC for TCEB then. This assumption will not be changed.
--> no code change required for exec. reader waiting for task ECB

Review Critical Modules for NP Work Units

IPW$$NU: All its services have been found ok, do not need NP mode. All exits and appendages covered already.
Review of IPWSEGM I/F routine at 'SEG00' and its critical steps:

1. entered PU under spooled partition task right from macro expansion
2. may conditionally BAL to 3800 logic module, which decides on its own, if prot. key 0 (and hence NP unit) is needed
3. at 'SEG80' get prot. key 0 for TIBFLAG5 update by OI - done automatically as NP work unit, as required for Supvr. Ctl. Block update
4. give up prot. key 0, hence enter PU work unit again
5. return to IPWSEGM macro expansion in PU mode
6. ---> no code change required

SEGMENT Macro: Extension runs as PU work unit under user partition task, enters $$BSGMNT transient by SVC 2, gaining automatically prot. key 0 and NP, and returns from transient with key ¬= 0, hence in PU mode, hence
--> no code change required.

Services

Resource Management

Resource management is responsible for the protection of serially-reusable resources (control blocks) against concurrent access by more than one task. Entry to the services is made by means of the macro instructions IPW$RSR (reserve resource) and IPW$RLR (release resource).

Reserve Resource: The reserve-resource service is entered when a VSE/POWER task issues an IPW$RSR macro instruction.

The resource lockword (bytes 28-31 of each resource control block) is examined. If the resource is not available (lock byte contains X'FF') the routine waits till it is available (by issuing IPW$WFL macro to task management). Task management requires that the lockword reside in pfixed storage, otherwise task selection might suffer a page fault. If the resource is available, ownership of the resource is established by storing the address of the TCB of the owning task in bytes 1 to 3 of the lockword.

If the resource to be exclusively reserved is either the DMB or the ACB, and the resource is not available, and it is a shared environment, the work-to-do ECB is posted in order to interrupt the T3 time interval.

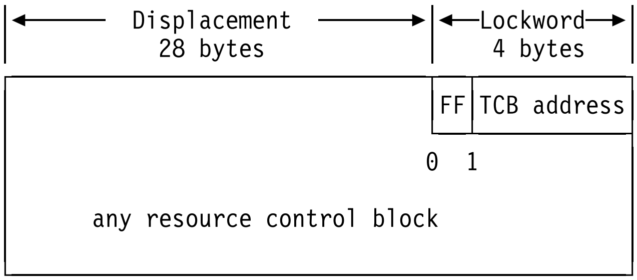


Figure 38. Resource LOCKWORD of a VSE/POWER Control Block

Release Resource: This service is entered when a VSE/POWER task issues a IPW$RLR macro instruction.

The resource lockword owner address is examined. If the task issuing the release request is not the resource owner the request is ignored. Otherwise, the lock byte in the resource lockword is set to zero so that the resource becomes available for use by any other task that may require it.

Real Storage Management

Real Storage Management controls the fixable storage area, whose maximum amount has been specified by the SETPFIX LIMIT=nnnk value for the VSE/POWER partition. Units of real Work Space (as requested by a task through IPW$RSW) are reserved and PFIX'd in this fixable area - and later released and PFREE'd again, when returned to the fixable area by an IPW$RLW request.

The storage control block (SCB), with page control table (isomorphic map of all pages in fixed area) and buffer control words (BCWs) are used to control the availability of pfixed address storage in the VSE/POWER partition (see Figure 39). The page control table consists of a bit-map whereby each page is represented by a bit. Bit positions with value 1 indicate that the corresponding page is fixed. Bit positions with value 0 represent pages which are either not yet fixed or explicitly freed via the PFREE macro instruction.

The SCB is locked during handling of the reserve/release request.

At VSE/POWER initialization time, the first and last page are fixed. The first BCW is placed in the first page at displacement X'18' and the last BCW is placed in the last two words of the last page. Each BCW is 8 bytes long and contains the length of the preceding buffer area and the length of the following buffer area; thus the BCWs are chained together by means of the length fields. If the buffer is in use, the length is stored in complement form (negative value). Since real storage is acquired in multiple of 32 bytes, the length fields contain the length divided by 32.

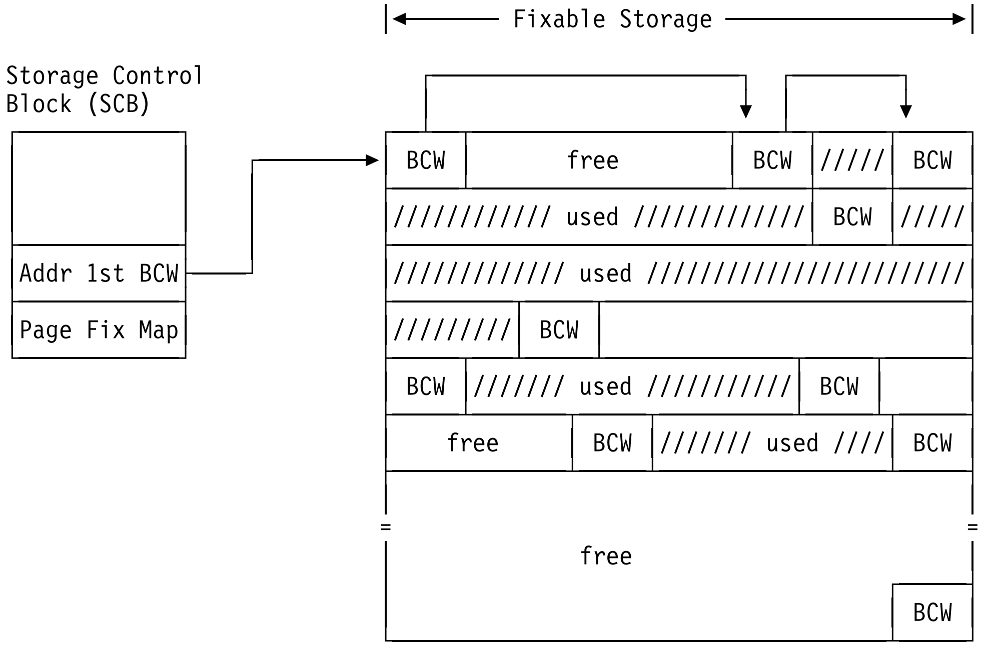

Fixable Storage
Storage Control Block (SCB)
Addr 1st BCW
Page Fix Map
BCW
free
BCW
/////
BCW
//////////// used /////////////
BCW
/////
///////////// used ///////////////////////
///////// BCW
BCW /////// used /////////// BCW
free BCW /////// used //// BCW
free
BCW

Figure 39. Storage Management Control Blocks Relationship

Reserve Real Work Space: The reserve-work-space service is entered when a VSE/POWER task issues a reserve work space (IPW$RSW) macro instruction. The BCW chain is scanned from the beginning to determine whether the required buffer space is available. If the required buffer space is not available in the already fixed pages, new pages are fixed (PFIX) to satisfy the request. Space is then allocated in the new page(s).

When buffer space is allocated by storage management, the BCW describing the free storage is updated to reflect that the following buffer space is in use and a new BCW is built following the acquired buffer area. The newly created BCW contains the address of the task requesting the buffer space and the length of the preceding and following buffer space either free or in use. If the requested buffer space is smaller than 4088 bytes, the buffer area is allocated in one page (no page crossing). If the system is running in /370 mode the virtual and optionally the real address are passed to the calling routine. If the system is running in virtual mode, the addresses passed are virtual addresses only.

If no buffer space is available and the requesting task has specified to wait, the task is put into wait state ($WFC) until work space becomes available. Additionally the operator is informed, via message 1Q59I, that the task is waiting for real storage. At each subsequent real storage post, which is done whenever real storage is returned to the real storage pool (by means of the IPW$RLW macro), the waiting task regains control and attempts to reserve the requested storage. This process continues until the work space request can be satisfied.

Release Real Work Space: The release-work-space service is entered when a VSE/POWER task issues a release work space (IPW$RLW) macro instruction. The buffer is cleared (binary zero) and the appropriate buffer control words are updated. If the page is no longer in use (all buffers are cleared) the page is freed (PFREE). Additionally the real storage ECB is posted to show that now real storage is available.

Real Storage Cushion: A short on storage cushion is held for authorized requestors whose function should complete even in short on real storage state. Authorized are currently:

- The PDISPLAY permanent command processor which wants to start a new task and needs therefore a PS-TCB (print status task).
- The print status task which needs real storage for the print status work area.
- The execution reader which needs real storage to process the execution account record after SVC 90/91.
- Reserve queue record ($RQS) which needs a register save area before calling allocate DBLK group with DMB locked already.

The cushion request is passed with IPW$RSW macro.

The real storage cushion is reserved during VSE/POWER initialization in IPW$$I7. IPW$$I7 contains a table (REALCUSH) where the amount for each cushion buffer is defined. For each buffer a IPW$RSW is done. The reserved real storage is marked as cushion element in the BCW and freed again via IPW$RLW. Whereupon it is marked free but is not linked to adjacent free buffers. Instead its original length is kept for authorized requests. Refer also to IPW$$I7 "Setup short on real storage cushion".

Exploitation of Fragmented Real/Virtual Storage: *Shortcoming of releases previous to VSE/POWER 5.1:* For many SOS (Short-On-Storage) tasks only the first will be posted for sure after a release storage request - others may continue waiting, in case the first task does not obtain its possibly big storage amount.

Improved design: Every SOS task records the 'storage state' of its last failing reserve attempt. With every release request the storage state (count 0-255) is incremented. Task selection of IPW$WFC will grant a reserve request to ALL tasks on that modified storage state.

Note: Leftmost ECB byte of real/virtual storage control block is used to maintain the storage count.

Virtual Storage Management

Virtual storage management controls the GETVIS storage allocated to the VSE/POWER partition. Work space in the GETVIS area for a task is reserved and released as requested by the calling routine. Storage management makes use of the subpooling possibilities to control storage allocation. The smallest unit of storage that may be reserved is 128 bytes.

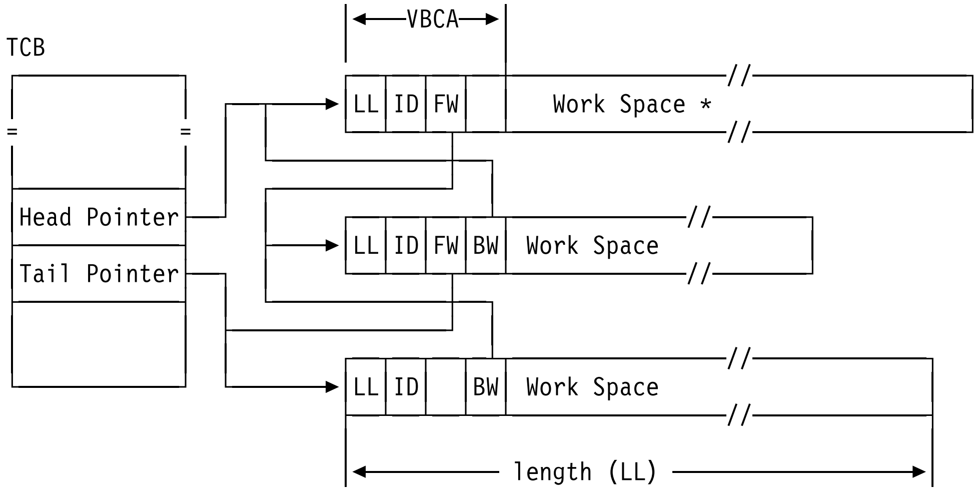
Reserve Virtual Storage: The reserve-virtual-work space service is entered when a VSE/POWER task issues a reserve virtual storage (IPW$RSV) macro instruction. The macro permits the caller to specify a pool type with the request. According to the type, the work space is obtained from the correct subpool within the GETVIS area. All pools are aligned on page boundary. If any subpool is empty it is automatically released by VSE/AF so that the GETVIS storage is available for other subpools if required.

The service routine rounds up the requested length of the work space to a multiple of 128 and selects the subpool anchor, according to the type specified by the caller. A GETVIS macro is executed to obtain the work space from the GETVIS subpool. If the caller demanded alignment of the work space on page boundary, the appropriate request is passed to VSE/AF.

If work space is available the routine allocates the storage area to the caller and initializes the work space header (also referred as virtual buffer control area). The header contains control information used by virtual storage management and precedes each acquired piece of storage.

The work space is chained as last entry in one of following queues:

- Own task queue (head and tail pointer are contained in task TCB)
- Other task queue (head and tail pointer are contained in TCB of other task)
- System queue (head and tail pointer are contained in the virtual storage control block or any other major control block).



```
Note:
 LL = Length of acquired work space (rounded up to next multiple of 128)
 ID = VSE/POWER assigned subpool identifier
 FW = Forward pointer, addressing next work space in chain
 BW = Backward pointer, addressing previous work space in chain
 *  = can be in different GETVIS subpools
```

Figure 40. Virtual Storage Relationship

The queue also determines the owner of the work space. All work space anchored to the task queue is automatically released when the task terminates. The virtual storage queues are double-threaded. Each header (VBCA) within the queue points to the next VBCA as well as to the previous VBCA. The queue itself is addressed by the head and tail pointer.

If enough work space is not available and the caller had specified WAIT=YES, which is the default, the task is put into wait state ($WFS) until work space becomes available. Additionally the operator is informed, via message 1Q85I, that the task is waiting for virtual storage. At each subsequent virtual storage post, which is done whenever virtual storage is returned to the GETVIS pool (by means of the IPW$RLV macro), the waiting task regains control and attempts to reserve the requested storage. This process continues until the work space request can be satisfied. If, however, WAIT=COND was specified and immediate termination is posted in the caller's task TCB, return is made to the calling task. In that case, the routine sets register 1 to zero to indicate that no work space was allocated.

If the work space request cannot be satisfied immediately and the caller did not elect to wait, the service routine sets the GETVIS return code in R0 and R1 to zero to indicate no work space available and returns to the caller.

Release Virtual Storage: The release-virtual-work space service is entered when a VSE/POWER task issues a release virtual storage (IPW$RLV) macro instruction. The routine removes the storage area from the virtual storage queue and frees the storage area by issuing the VSE/AF FREEVIS macro. If the storage area to be freed is not a member of the task virtual storage queue, the head and tail address must be provided by the caller. The routine posts the virtual storage ECB to show that storage is now available and returns to the caller.

Unchain Virtual Storage Element: The unchain-virtual-storage service is entered when a VSE/POWER task issues an unchain virtual storage (IPW$UNV) macro instruction. The routine performs two functions:

1. A specific storage element, addressed by register 1, is removed from the specified virtual storage queue and chained to another queue.
2. The first element of a specified queue, if any, is unchained and chained at the tail of the issuing task virtual storage queue. The address of the element is returned in register 1. If the queue is empty, register 1 is set to zero.

Message Service

Local Message Service: See also “Message Handler Overview” on page 143 and “Message Reference” on page 401. The local message service is invoked by:

- a IPW$WTO or IPW$WTR macro instruction issued by the calling routine. It performs a console write operation or a write operation followed by a read operation, defined by information supplied by the calling routine in the message request word located in the TCB. (See Figure 49 on page 136). The message request word and reply request word contain the addresses of message and reply areas of the calling routine.
- a IPW$GAM macro instruction, used to obtain a message from the message definition module IPW$$MM, which contains most local and remote messages. It performs one of the following functions:
  - Move message into user-supplied area
  - Return message address
  - Write message to central operator
  - Add message to remote message queue

There is a message control block (MMB), which is locked for the duration of the operation. It contains a channel program and CCB for issuing the message if the WTO/WTOR should fail, the message output area and the reply input area. This resource is used to serialize parts of the IPW$$MS code.

Remote Message Service: The remote message service is also used to support remote message handling by the use of the IPW$RMS macro instruction. It performs one of the following functions:

- Add to remote message queue
- Delete from remote message queue
- Get message from remote message queue
- Get first/next ALLUSER type message
- Add message to ALLUSER type message queue
- Delete ALLUSER type message(s)

The function to be performed is indicated in the function indicator byte supplied by the caller. Similar to the local message control block, there exists a remote message control block, which is locked for the duration of the operation. The control block, which is set up at VSE/POWER initialization time, contains among others, the caller's registers at entry point of the routine.

Nodal Message Service: This routine is entered when a VSE/POWER task issues a IPW$ICS REQ=ADD macro instruction. The routine locks the remote message control block for the duration of the operation. The routine performs one of following functions:

- Adds a message which is already in nodal message record (NMR) format at the tail of the message queue of the appropriate node control block. The destination of the NMR is defined in the NMR itself.

  Note: The NMR is used to send messages and commands within the network.

- Builds a nodal message record and adds this record to the message queue of the appropriate node control block. The message id and the target node and remote name are supplied by the caller in the message request word located in the TCB of the calling task.

Note: The following path determination scheme is used to transmit the nodal message record:

1. If a connection exists to the prime (adjacent) route node, the NMR is queued on this NCB.
2. If no such connection exists but a connection exists to the alternate route node, if one was specified, the NMR is queued on this NCB.
3. Otherwise, the NAT table is examined if the prime or alternate routing node is connected to another system, participating in the shared spooling complex. If so, the NMR is passed to the slot manager to be forwarded across shared spool.

Notify Message Service Notify Service is entered when a task issues a notify macro (IPW$NTY) instruction. There are two types of notify services, which can be selected by specifying or omitting the QCM operand of the notify macro.

Specifying the QCM operand: The QCM operand of the notify macro is used to store the completion message generated by VSE/POWER after job completion into a piece of virtual storage for later retrieval by a user written application program. The logical address of this piece of storage is defined by the combination of the XPCC application ID and the Spool-access support user ID which are specified when the job is submitted to VSE/POWER. Later at the time of retrieval, the combination of application- and userid ID has to be specified again by the message retrieving program. The messages can be retrieved from any user written program by means of the GCM service of VSE/POWER's Spool-access support.

This type of notify service is applicable only to tasks which process a job which has been submitted to VSE/POWER via the Spool-access support interface. The job must be submitted with the 'queue completion message' option (SPLGFB1 equated to SPLGF1QM) specified in the SPL.

In order to invoke the service QCM=YES must be coded in the macro. The service is used by an execution reader task.

In order to make program driven evaluation of completion messages easier the messages are stored in fixed format. The layout of one message is given by the Spool-access support macro PWRSPL in DSECT JCMDS. The messages may be passed over the network. Therefore a message is always converted to nodal message record format at the time of creation. In the following such a message is called a fixed format Nodal Message Record (f.f.NMR).

Job Generation Messages: Module IPW$$XWE generates fixed format generation messages, if the Spool-access support user requested the generation by specifying SPLGF1QQ instead of SPLGF1QM. Whenever a DISP=I operand is encountered in a * $$ PUN statement, and a new VSE/POWER job is generated such a message is created. A fixed format job generation message is accessable by DSECT JGMDS.

Job Generation Messages and Job Completion Messages are called "Job Event Messages" in this document.

Local Message Queuing: If the job entered the system with with the 'queue completion message' option, VSE/POWER flags the job with a bit (QRO2QCM) defined in the queue record. IPW$$XRE checks the flag and decides whether the job completion message has to be issued by means of the IPW$NTY macro with or without the 'QCM=YES' operand in order to queue the message to a specific message queue. If the macro is issued with the QCM operand, the message is queued to a specific message queue. This message queue is defined by the Spool-access user by the Spool-access application Id and the Spool-access user Id when the job is submitted. Then the The macro calls the notify service of IPW$$NU, from where IPW$$MS is called. The job completion message is converted from internal format to a fixed format NMR and returned to IPW$$NU. Finally, if the message is destined for the local node, IPW$$NU calls IPW$$NS where the message is queued to the the message queue (Figure 41).

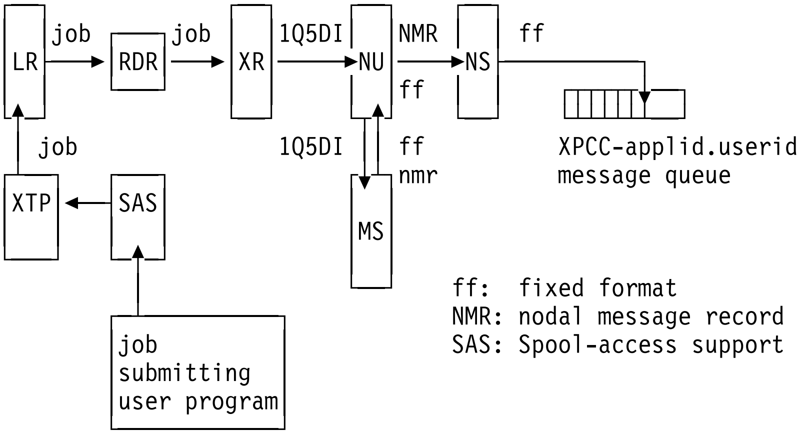


Figure 41. Local Job Submission and Notification Message Queuing

Local Shared System: In contrast to local message processing the nucleus calls IPW$$MS again rather than passing the message to IPW$$NS after the fixed format NMR has been created. Then IPW$$MS provides for that the fixed format NMR is added to the Queue Control Area (QCA). The timer task (IPW$$TI) at the target system retrieves the NMR from the QCA and gives control to IPW$$MX, which calls finally IPW$$NS. IPW$$NS queues then the message to the message queue (Figure 42).

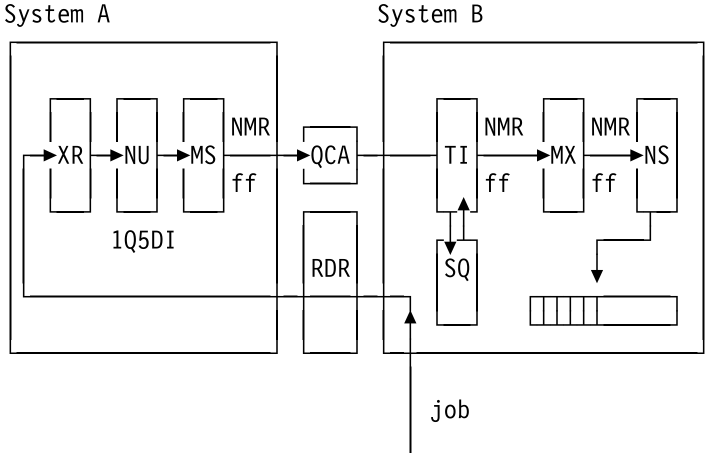


Figure 42. Shared System Notify Message Queuing

Networking and Message Queuing After the f.f. NMR has been created at the message transmitting system, the f.f. NMR is anchored in the NCB and the line driver is posted. The message is then transmitted by IPW$$NT to the next node. At the receiver's side, the NMR is received by IPW$$NR, and IPW$$NR2 respectively, where message distribution IPW$$MX is called, which either

1. passes the NMR to the nucleus, which then calls IPW$$NS to queue the message, if the final system-id is reached (Figure 43) or
2. stores the NMR into the QCA via IPW$$SQ (build slot routine), when the message has to be passed to another system-id which participates in the shared system complex. The timer task at the final system passes the NMR to IPW$$NS, where the message is then queued (Figure 44 on page 131).

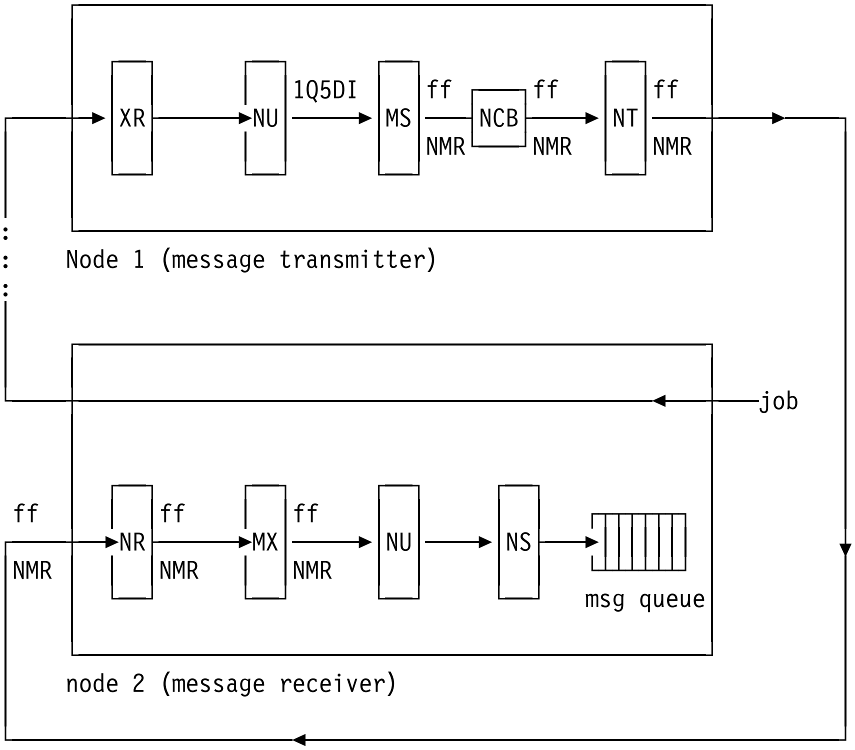


Figure 43. Job Completion Message Queuing at Receiving System

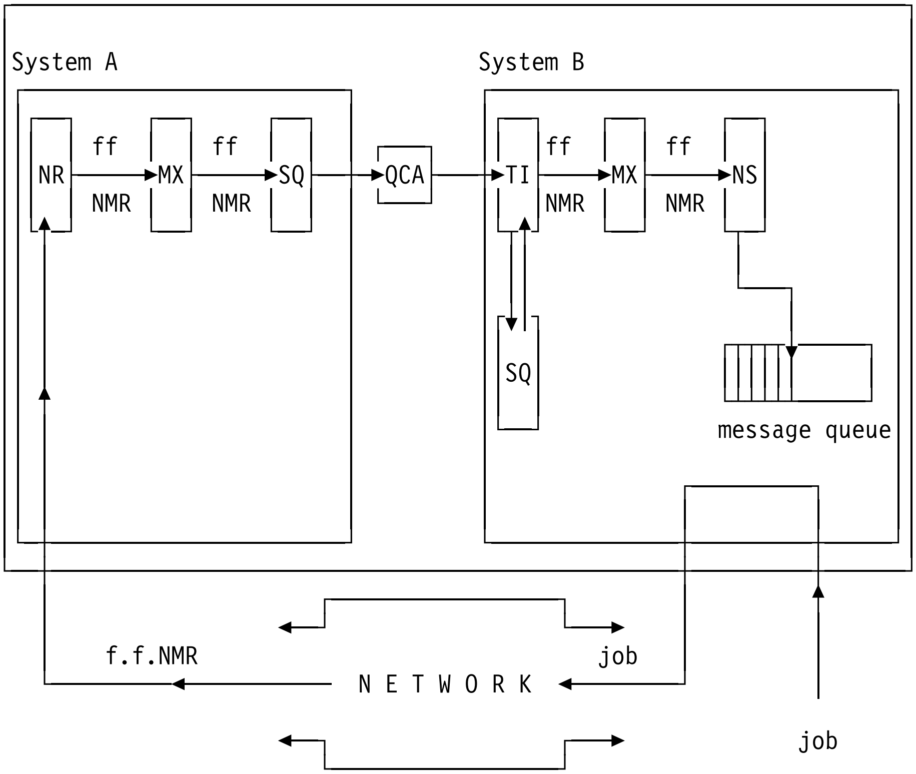


Figure 44. Job Completion Message Queuing at Shared Receiving System

Shared Processing System: The next scenario shows the job and completion message flow when the job enters at node A but is processed on node B which is a shared VSE/POWER system. The job is received on system B1 but is processed on system B2. The resulting job completion message is converted to fixed format in IPW$$MS and stored to the QCA by means of the IPW$IQS service. The timer task of system B1 then picks up the message from the QCA, and queues the fixed format message to the NCB by means of the IPW$ICS service macro call. Then the message is transmitted by the network transmitter to node A. At node A, the fixed format message is received and finally stored in the fixed format message queue (Figure 45 on page 132).

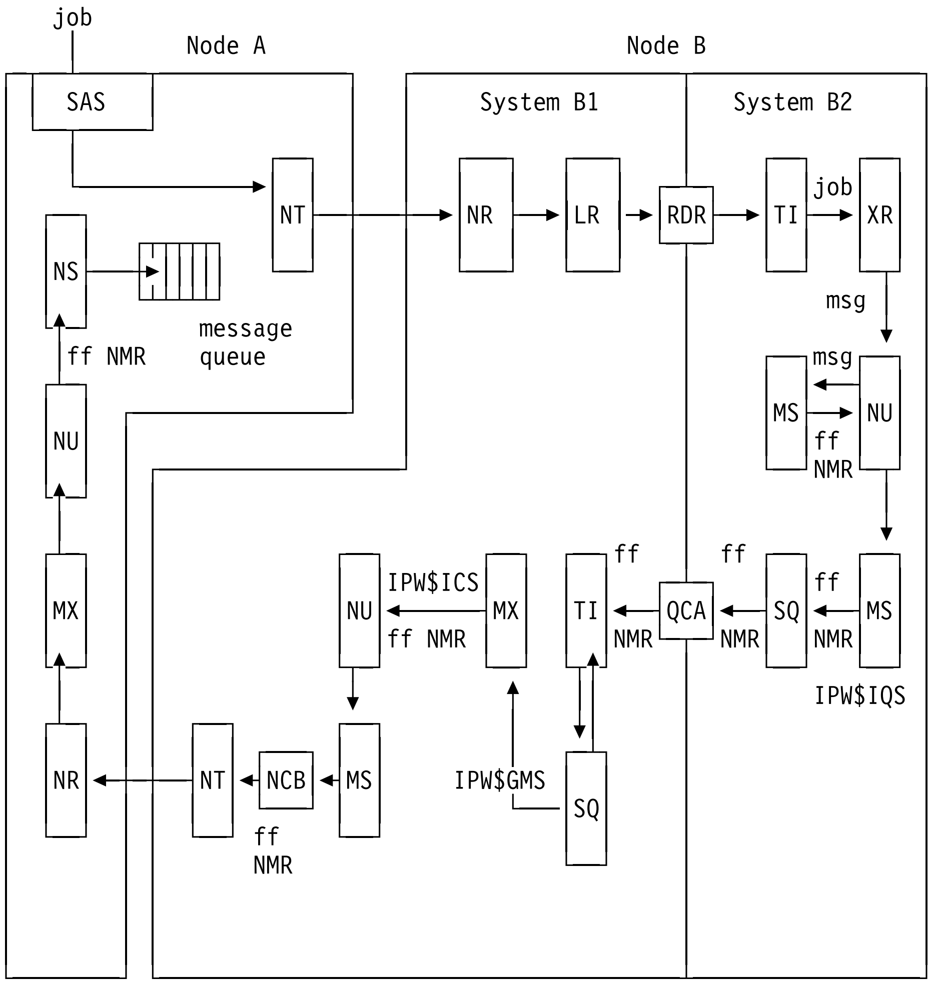


Figure 45. Message Flow with Shared Processing System and Network

Message Handling at Local Node and System: The following summarizes all processing of f.f.NMR at the local VSE/POWER node and system.

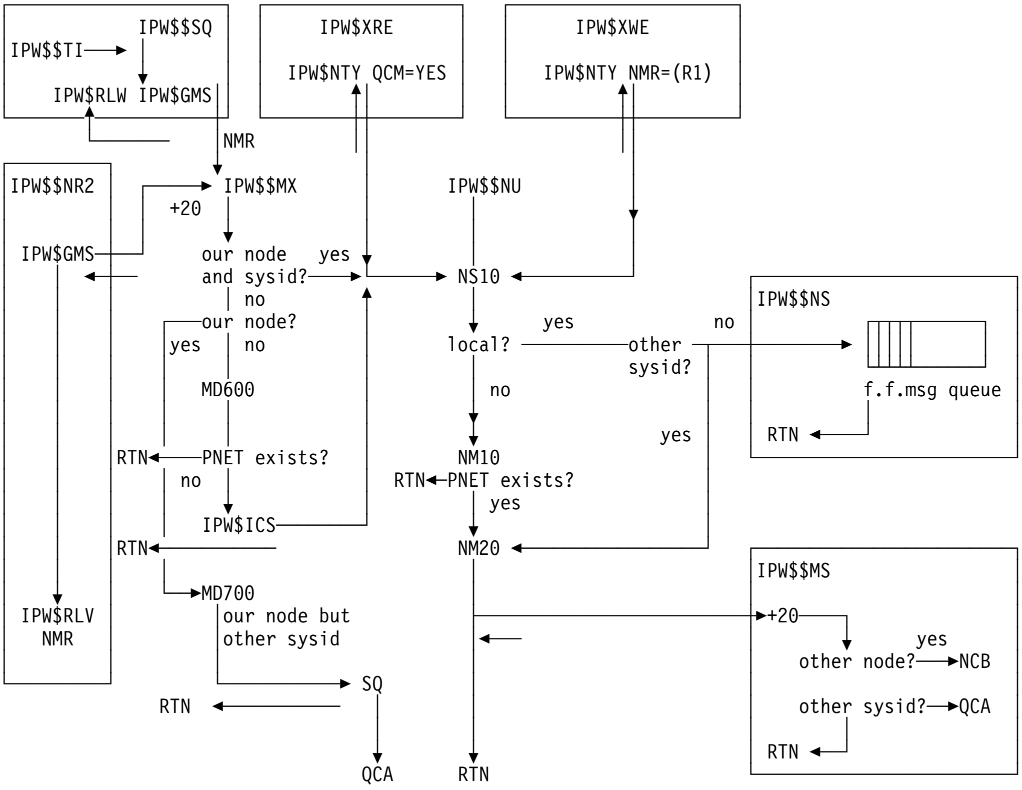


Figure 46. Job Event Message Queuing. Sources of Fixed Format Messages and Message Distribution

Final Message Queuing: A f.f. NMR is finally stored in the GETVIS area of the VSE/POWER partition by module IPW$$NS. The piece of virtual storage for one message is logically represented by an entry of the - as it is called - 'fixed format message queue'. For each application- and user-ID combination one unique message queue exists. The capacity of this message queue is defined by the SET JCMQ autostart statement. A queue size in the range from 0 to 99 entries can be specified. The specified size is stored in the Communicator Information Block 2 (CI2). One message queue is identified by an Application Communicator Information Element (ACIE), that is, for each single ACIE one fixed format message queue exists. The ACIEs are chained. The first ACIE of the chain is pointed to by the CI2. The CI2 is created at initialization time by the cross partition master task and is located in fixed storage. One single ACIE has its own fixed format message queue associated. ACIEs and the fixed format message queues are located in virtual storage (Figure 47 on page 134). The ACIE pointer of the last ACIE in chain contains X'00'.

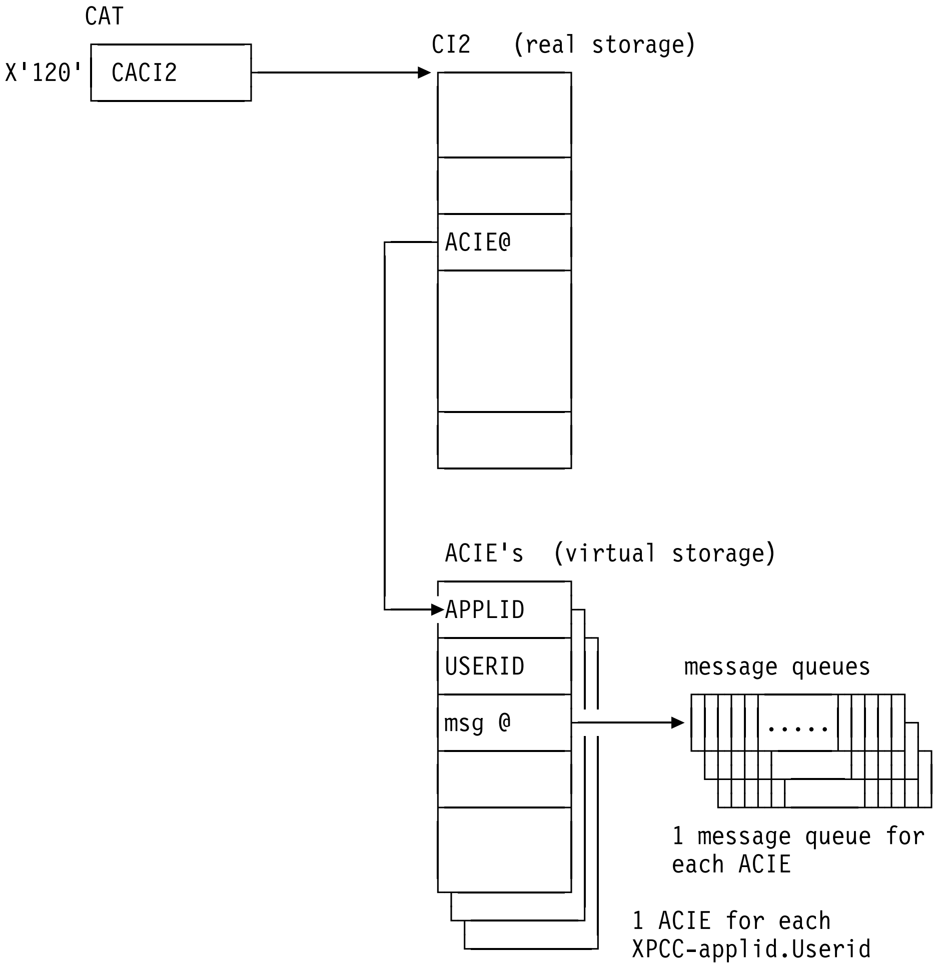


Figure 47. Control Block Relationship for Communicator Information Block 2

Module IPW$$NS has the following tasks for the 'QCM' service

1. conversion of the f.f. NMR to the message queue format
2. creation of an ACIE if required, and
3. appending the converted f.f. NMR to the message queue of the relevant ACIE.
4. posting a cross partition user task waiting for job event messages.

Message passing and queuing works also for job generation messages. Such messages are generated by an execution writer task when a new job has been created by means of a * $$ PUN DISP=I statement.

Job Completion and Job Generation Messages are called Job Event Messages in the following text.

Conversion from f.f. NMR format to message queue format is shown in Figure 48 on page 135.

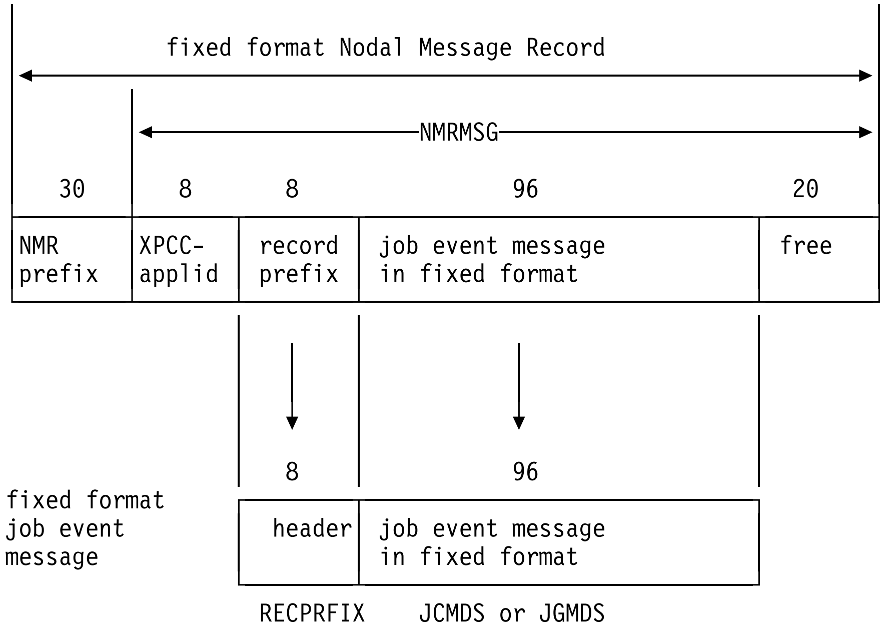

fixed format Nodal Message Record
NMRMSG
30
8
8
96
20
NMR prefix
XPCC-applid
record prefix
job event message in fixed format
free
8
96
fixed format job event message
header
job event message in fixed format
RECPRFIX
JCMDS or JGMDS

Figure 48. Conversion from f.f. NMR to Message Queue Format

The layout of one queued message is given by the DSECTS JCMDS or JGMDS, and RECPRFIX, both contained in VSE/POWER's PWRSPL macro.

Locking Mechanism In order to avoid uncontrolled access to the message queues by different message queuing and de-queuing tasks a locking mechanism is established for the CI2. This is done by the IPW$RSR macro and by the IPW$RLR macro, respectively. The nucleus IPW$$NU reserves the CI2 before the notify service (IPW$$NS) is called. After the message is queued in IPW$$NS, the task returns to the nucleus where the CI2 is then released. Locking of the CI2 is checked by IPW$$NS when a message is to be queued as well as by IPW$$XTM, when a message is retrieved by the Spool-access GCM service.

Omitting the QCM operand: This routine is entered when a VSE/POWER task issues a notify (IPW$NTY) macro instruction without specifying the QCM operand. If the message is already in NMR format, the routine queues the message to the VSE/ICCF notify message queue. In all other cases the routine acts as a distributor. Depending on the destination, the routine directs the messages to:

- Local operator
- Any remote operator locally attached
- Any local VSE/ICCF user
- Any user on another node
- Any subsystem running on the local system

The message id, target node and remote name, if applicable, are supplied by the calling task in the message request word and in register 0.

Note: No information is passed back to the calling task on whether or not the message was successfully queued.

The variable portions of the message text are converted to indicate information pertinent to the specific task or queue entry when combined with this message.

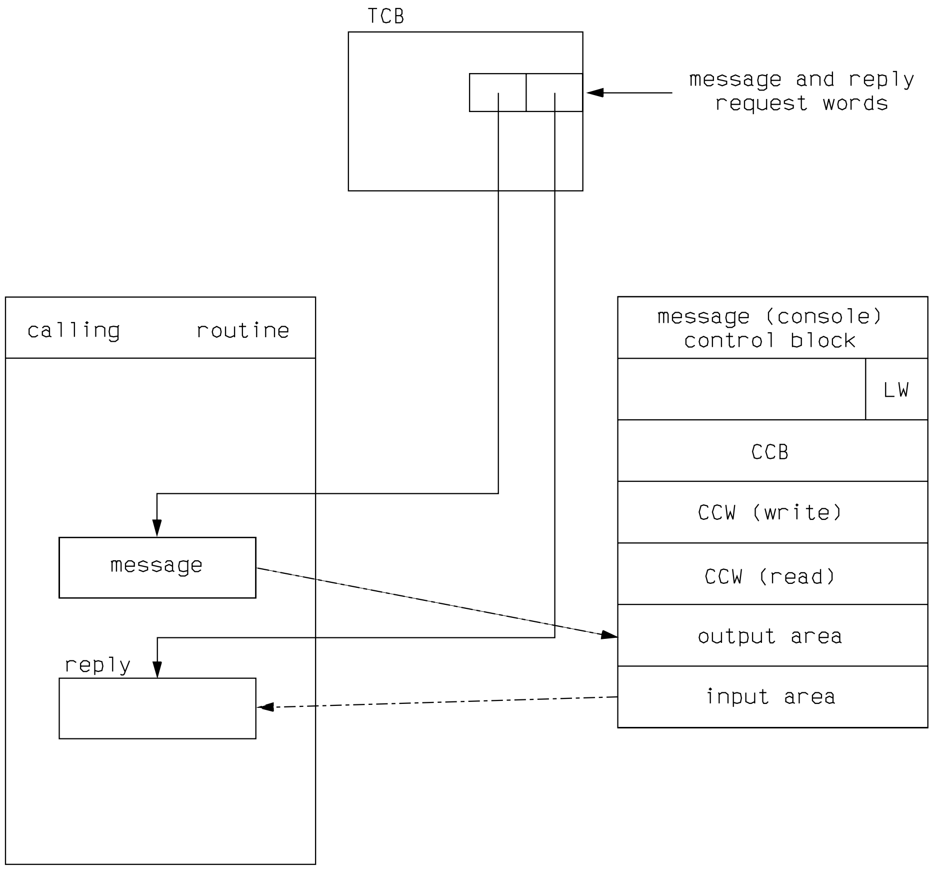


Figure 49. Message Service Control Block Relationship

Queue File Server

The Queue file server consists of the following set of routines:

- Get queue record
- Modify queue record
- Write back queue record to disk

Get Queue Record: The service routine is invoked by means of the IPW$GQR macro instruction. On entry, register 1 addresses the I/O request word, supplied by the calling task. The I/O request word defines the queue record supposed to be obtained by means of the relative queue record number and the queue record area address. See Figure 50 on page 138 for the layout of the control block. The queue file MCB is locked for the duration of the processing.

The routine calculates the relative address of the compartment holding the appropriate queue record by multiplying the relative queue record number by the compartment size (currently 384). The VIO/GETVIS-MOVE subroutine, contained in the VSE/POWER nucleus, is then called to move the queue record into the area supplied by the calling task.

Modify Queue Record: The service routine is invoked by means of the IPW$MQR macro instruction. On entry, register 1 addresses the I/O request word, supplied by the calling task. The I/O request word defines the queue record to be updated. The update of the queue record is performed only in the storage copy of the queue file. The queue file MCB is locked for the duration of the processing.

The routine calculates the relative address of the compartment holding the appropriate queue record by multiplying the relative queue record number by the compartment size (currently 384). The VIO/GETVIS-MOVE subroutine is then called to move the queue record addressed by the I/O request word into the appropriate compartment in the VIO or GETVIS space.

When running shared, the 'refresh' flag of the own SYSID is set in the DMB (refresh table) recording that the appropriate queue record block is modified. This causes that the timer task will write back all changed queue record blocks to disk at the end of the T1 interval.

Write Queue Record: The service routine is invoked by means of the IPW$WQR macro instruction. On entry, register 1 addresses the I/O request word, supplied by the calling task. The I/O request word defines the queue record to be written. The update of the queue record is done both in the storage copy of the queue file and the queue file on disk. On the contrary to the IPW$MQR macro, the macro must be coded when the status of a queue record has been changed. The queue file MCB is locked for the duration of the processing.

The routine calculates the relative address of the compartment holding the appropriate queue record by multiplying the relative queue record number by the compartment size (currently 384). The VIO/GETVIS-MOVE subroutine is then called to move the queue record, addressed by the I/O request word, into the appropriate compartment in the VIO or GETVIS space.

The IPW$WTQ macro is issued in order to write the queue record block back to disk and if running shared the refresh bits are set to inform the other systems that the appropriate queue record block was changed; the 'refresh' bit representing the own system is turned off. The queue record block number is calculated by dividing the relative queue record number by the number of queue records per block.

VIO/GETVIS-MOVE Subroutine: This subroutine is used to move data from/to the VIO or GETVIS space, depending on where the copy of the queue file on disk actually resides in storage.

If it resides in VIO space, the routine makes use of asynchronous processing by using VIO POINT to address the appropriate VIO block prior to moving data from/to the VIO space, which is performed in chunks of VIO blocks. First a check is made, if the appropriate VIO block is addressable. If not, a VIO POINT macro is issued followed by a wait (IPW$WFC) before the move is done. If the caller's supplied length is greater than the VIO block size, the next VIO block is made addressable, the to and from addresses are adjusted and the next piece is moved. This process continues until the entire area is moved to/from the VIO space.

If the queue file copy resides in partition GETVIS space, the requested move RBA address is simply added to the start address of the queue file, and the from/to move request is done with a single Move-Long instruction. For that instruction the addressing mode is switched from Amode-24 to Amode-31 and back again, because the Q-file may reside beyond the 16MB line (when sufficient partition ALLOC has been provided). Any page fault occurring during execution of the MVCL is handled by the Page Fault Pre-Processor, which records the Amode-31 using a TCB flag, so that later at re-dispatch, the task can resume the MVCL with correct addressing mode.

Note: Any error condition returned by the supervisor as result of a VIO POINT SVC will cause VSE/POWER to terminate abnormally with message 1QB5I. Any move request from/to a queue file entry outside the copy in VIO or GETVIS will also cause VSE/POWER to terminate abnormally with message 1QZ0I RC=21/23.

Disk Service

Disk service is invoked by IPWRDQ, IPWWTQ, IPW$RDD, or IPW$WTD macro instructions issued by the calling routine. It reads or writes records to the queue file or the data file defined by the information supplied by the calling routine in the I/O request words in the TCB. See Figure 51 on page 140.

The I/O request word contains the relative DBLK number, the virtual address of the logical data area (LDA) and optionally, the length of the data area used. If no length is specified, the I/O will be done in the length of a DBLK. For a queue file I/O, the I/O request word contains the relative queue record block number and optionally the address of the storage area and its length (master record only).

```
   ┌───────────────┬───────────────┬───────┬───┬───┐
   │ R   R   R   R │ A   A   A   A │ L   L │ C │ F │
   └───┴───┴───┴───┴───┴───┴───┴───┴───┴───┴───┴───┘
   0               4               8       10  11
where:
    RRRR –   Relative queue record block or DBLK number
    AAAA –   Storage area address
    LL   –   Length of area to be read/written
    C    –   Read/write code for queue/data file
    F    –   Flag byte
```

Figure 50. I/O Request Word

There is one MCB for every queue or data file extent, which is locked for the duration of the operation. It contains the CCB and skeleton channel program, which is appropriately initialized for each I/O operation. Each MCB is equipped with an I/O area in the length of a queue record block or master record, whatever is larger, for the queue file or a DBLK for the data file. This I/O area is fixed, anchored to the MCB, and used for each I/O. If the I/O area occupies more than one page and running in /370 mode, an IDAL list is built at VSE/POWER initialization time and the channel program is updated to reflect the IDAL list.

The disk service routine performs the following functions:

- Read/Write operation queue file:

  1. For a write operation, the 'VIO/GETVIS-MOVE' subroutine is called to move the queue record block from VIO or GETVIS space into the I/O area. The queue record block address is calculated by multiplying the relative block number by the queue record block size. If the master record is to be written, it is moved from the caller's provided area into the I/O buffer. If the queue record block to be written is marked “inaccessible” in the defect queue record block map, the actual I/O operation is suppressed.

  2. The seek address is calculated from the queue record block number by dividing it by the number of blocks per track on the particular device, obtaining the relative track number within the extent (the remainder + 1 is the relative record number on the track) and dividing the relative track number by the number of tracks per cylinder on the particular device type. The absolute track number is obtained by adding the begin extent track number. This value is then divided by the number of tracks per cylinder. The remainder is the track number; the absolute cylinder number is obtained by adding the begin extent cylinder number to the quotient. Both the number of blocks per track and the number of tracks per cylinder are maintained in the MCB. If the queue file resides on a FBA device, the relative FBA block number will be calculated by multiplying the relative queue record block number by the unit of transfer (number of FBA blocks per queue record block).

  3. Then the pre-built channel program is completed with seek address and set sector value, if applicable. Next, the channel program is executed and a wait is performed (IPW$WFC). After com-

pletion tests for wrong length and unrecoverable I/O errors are performed. If such an error is encountered, the I/O error handling routine is called.

4. For a read operation, the I/O area is moved into the VIO space by invoking the 'VIO Move' subroutine. If the master record was read in, it is moved into the storage area, supplied by the caller, instead.

If the queue record block to be read is marked “inaccessible” in the defect queue record block map, the actual I/O operation is suppressed and the appropriate VIO or GETVIS space is initialized with "B" signalling "bad" queue record block both in VIO/GETVIS area and for queue file recovery.

Note: When a system operates with an “in-storage” queue file only (after queue file write I/O error, which could not be repaired), all queue record block and master record read/write I/Os will be suppressed.

- Read/write operation data file:

  1. The data file MCB chain is scanned to locate the MCB, the DBLK to be read in or written belongs to. If the specified DBLK number is outside the total range

     – message 1QZ0I with reason code 1 is issued and VSE/POWER is abnormally terminated, provided it is still in the initialization period.

     – an IDUMP in flight is taken and the requesting task is handled by IPW$$TR as if an I/O error had occurred on the data file, indicated by message 1Q6GA or 1Q6HA or 1Q6KA.

  2. The seek address is obtained by first calculating the DBLK number relative to the begin of the extent. The following formula is used:

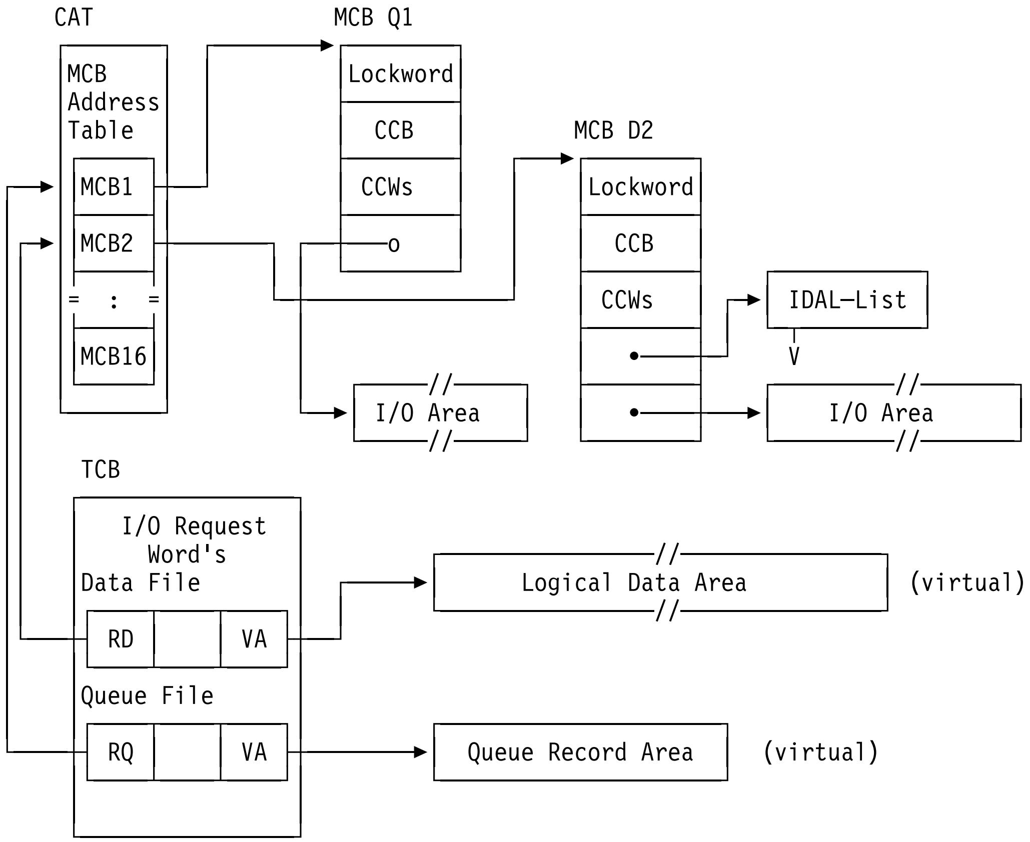
     ```
     relative DBLK number within extent =
                 DBLK number - DBLK number of first DBLK in extent
     ```

     This value is then divided by the number of DBLKs per track obtaining the track number relative to the begin of the extent. The remainder + 1 is the absolute record number on the track. The absolute track number is obtained by adding the begin extent track number (this value is saved at VSE/POWER initialization time in the MCB). The absolute track number is then divided by the number of tracks per cylinder yielding the conventional cylinder, track and record address.

     If the extent resides on an FBA device, the appropriate FBA block number is calculated by multiplying the relative DBLK number by the unit of transfer (number of FBA blocks per DBLK) yielding the relative FBA block number.

  3. *Read Operation:* The channel program is completed with seek address and the I/O is started. After the I/O completed, the contents of the I/O buffer is moved into the logical data area, supplied by the calling task in the length supplied by the caller. Any page faults, which might occur while moving the data into the virtual, logical data area are handled by the VSE/POWER page fault handler and do not impact other VSE/POWER tasks running.

     Write Operation: The logical data area, supplied by the calling task is moved into the I/O area in the length also supplied by the caller. The channel program set up at VSE/POWER initialization time is then completed with the seek address or locate word (FBA only) and the I/O is started using EXCP real. Once the I/O is started, immediate return is made to the caller, unless the caller requested to wait for I/O completion. In the first case, the I/O completion is checked the next time an I/O is done for the same data file extent.



Note:

```
  RD :  Relative DBLK number
  RQ :  Relative Queue record number
  VA :  Virtual address
```

Figure 51. Disk Management Control Blocks Relationship

Tape Service

Tape service is invoked by IPWWTT, IPWRDT, or IPW$CTT macro instructions issued by the calling routine. It reads or writes records to tape file, or performs a tape control operation defined by information supplied by the calling routine in the tape control block (TBB). The TBB is associated with the tape device and contains the skeletal channel program. See Figure 52.

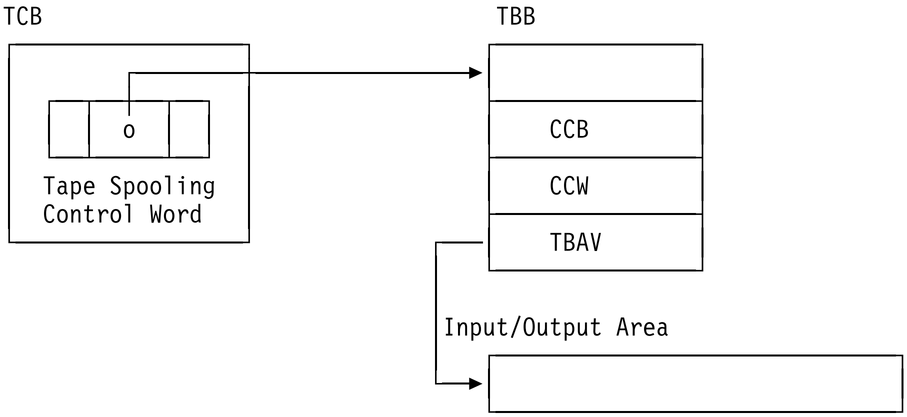


Figure 52. Tape Service Control Blocks Relationship

Timer Service

Timer service is invoked by the IPW$RDC macro instruction. It issues a GETIME standard macro instruction to obtain the time of day in packed decimal format. Also, the date field in the master record is updated with the value stored in the partition communication region.

Interval Timer Service

The Interval Timer Service provides an interface between VSE/POWER tasks and the standard VSE/AF timer facilities. It allows multiple task intervals to be active while maintaining only one VSE/AF timer interval through the SETIME macro. It provides notification for tasks on completion of intervals and remaining time cancellation. The user of the VSE/POWER interval timer service must provide a unique timer queue element (TQE) for each interval that is to be simultaneously active.

To begin an interval the VSE/POWER task executes a IPW$STM macro, which requests storage for the TQE, if not already present, formats the TQE and then invokes the interval timer service routine. During the interval, the TQE is chained to other TQEs in expiration time sequence. The SETIME macro is issued, if the first TQE is not the TQE currently represented by the last executed SETIME macro, or if there is no timer interval currently active for VSE/POWER. When the interval expires, the TQE is removed from the TQE chain and the task is posted.

When an active interval is to be terminated, the requesting task issues a IPW$STM CANCEL macro. The TQE is then removed from the active chain.

When VSE/AF recognizes the end of a timer interval set by VSE/POWER, control is given to the interval timer exit routine. (Linkage to that routine has been set up at VSE/POWER initialization time.) The exit routine records the fact that no timer interval is active and posts the VSE/POWER master ECB and sets VSE/POWER dispatchable. Whenever the VSE/POWER task dispatcher gets control, it checks if a timer interval is expired. If so, the interval timer routine is invoked which de-queues expired TQEs and posts associated tasks for work.

Validation Service

During VSE/POWER initialization in IPW$$I7 system-related boundary information (LTA boundaries, system GETVIS start and start of shared area) are obtained from the supervisor and saved in the CAT. If an address is found which is not within the allowed limits, boundary information is retrieved anew, because meanwhile the storage layout may have been altered by the operator. Validation service is invoked by the IPW$VDA macro instruction. The data address and its associated length which are provided in the user-supplied channel command word and the address of the CCW itself are examined to ensure that they relate to a data area that the user is allowed to access.

The user is allowed to access the user's partition, the logical transient area, and the shared virtual area, for read, write, or control operations. This is illustrated in Figure 53. If the validation fails VSE/POWER obtains the partition boundaries again via the EXTRACT macro to get updated information about it since it might be possible that the operator changed the allocation for the partition while it is under control of VSE/POWER.

	User Partition (including Dynamic Partition GETVIS Area)	LTA	SVA
DATA AREA	Valid	Valid	valid, if write or control
CCB (not validated)	Valid	Valid	Invalid
Channel Program (CCW)	Valid	Valid	Valid

Figure 53. Areas Checked by Validation Service

This validation routine is used only by the modules processing the execution tasks, which are IPW$$XRE and IPW$$XWE. Note that the CCB is neither validated by this validation routine nor the calling modules.

As the validation routine accesses data in the user partition (for example the CCW operation code, the address and length of the data), the validation routine uses access registers if running with an ESA-supervisor. In this case, the validation routine assumes that the access-register mode is set on by the calling routine.

The validation routine also passes an error code to the calling routine, if the CCB indicates that Format 1 CCWs are to be processed.

Remote Service

Remote service is invoked by the IPW$SRM macro instruction. Depending on the option specified, the bit representing the remote id is either turned on or off in the remote bit mask. The remote bit mask indicates which remote users are signed on at any time.

Get Trace Entry

This routine is invoked by the IPW$GTE macro instruction. The routine allocates a trace entry from the VSE/POWER trace table and returns its address in register 1 to the caller. If the current trace area is filled, the routine swaps to the alternate trace area and if trace logging was requested, a IPW$IAS TYPE=SERVICE macro instruction is issued to dump the filled trace area to the VSE/AF dump library, assigned to the VSE/POWER partition. Figure 54 on page 143 shows the two trace areas and how they are used.

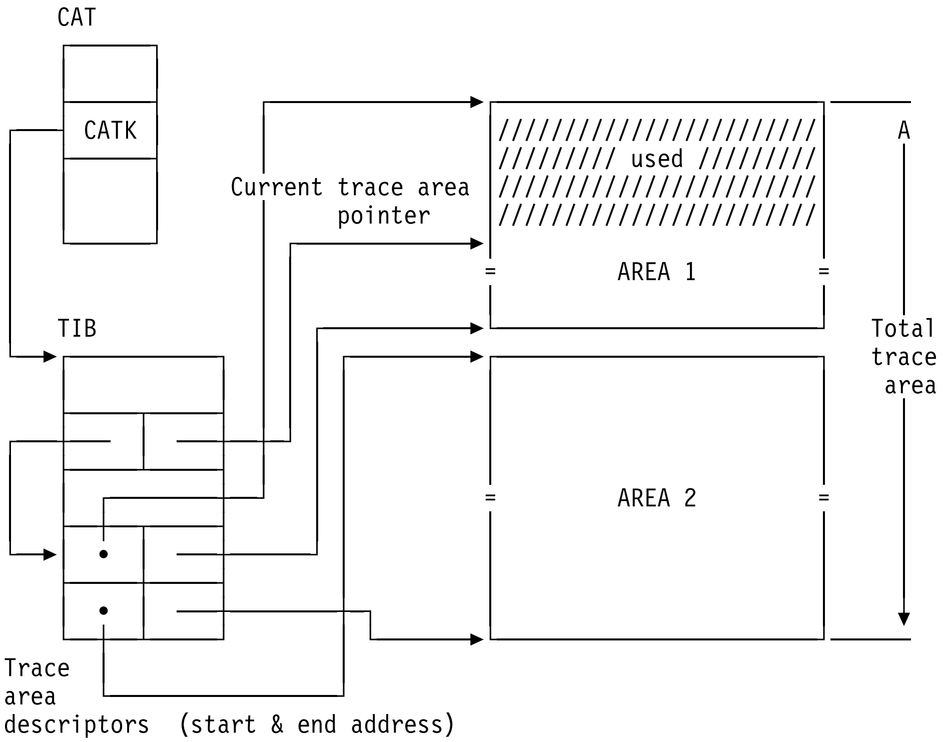

CAT
CATK
Current trace area pointer
////////// used //////////
AREA 1
A
Total trace area
TIB
AREA 2
Trace area descriptors (start & end address)

Figure 54. Trace Service Control Block Relationship

Switch NP/PA Mode Service

This routine is called by the IPW$TDM macro request and finally entered via the CAT service branch table entry 'PN00'. It acts upon request type NP/PU corresponding to passed R1=01/00 as follows.

1. If Turbo Dispatcher not activated, ignore request.
2. If all session non-parallel (default), that means CAF4WKNP=ON, then ignore request.
3. If INIT task requesting 'PU' mode, ignore request.
4. If selected PNET SNA or all RJE/SNA tasks (must always run 'NP') call for 'PU' mode, then ignore request.
5. If desired mode is already active, bypass TDSERV.
6. Request TDSERV FUNC=SWITCHNP/PU according to passed R1 value.
7. Handle TDSERV failure.
8. Set TCF16NP according to desired and acquired mode.

Miscellaneous Tasks and Functions

Message Handler Overview

This phase (IPW$$MS) handles local, remote as well as nodal message requests. See also “Message Service” on page 127 and “Message Reference” on page 401. It is called by the message service routine in the VSE/POWER nucleus whenever an IPW$GAM or IPW$WTO or IPW$WTR macro (local), or an IPW$RMS macro (remote), or an IPW$ICS REQ=ADD macro (nodal) is issued.

Local Message Request Information about the message to be issued is supplied by the calling routine in the message request word of the TCB. The message length is examined and, if necessary, truncated to the maximum of 132 characters. If the message is in NMR format, the originating node name and/or user/remote id are put in front of the actual message. The message text is scanned to determine whether any message modification is necessary. If so, the message text is modified in the appropriate modification routine. This is done by issuing the IPW$GMS TYPE=SUB macro instruction, which expands into a linkage to the IPW$$MX module. Afterwards the message text is squeezed if the text contains two or more consecutive blanks. A console write operation (for an IPW$WTO macro), or a console write operation followed by a read operation (for an IPW$WTR macro), is then performed. For PUTSPOOL, GETSPOOL, and CTLSPOOL processing, the first 60 characters of the message text are placed in the user's buffer area at a displacement offset of 28 bytes.
If the message is issued by a cross-partition (SAS) task or on behalf of such a task, the message is queued at the tail of the message queue anchored to the work area of the task concerned. These messages are then passed to the cross-partition user whenever appropriate. Certain critical messages, such as action-type messages, are also sent to the system operator.

Local Message Request - Support for VSE Macros WTO/WTOR/DOM: Usually local messages are issued via a VSE WTO macro (or WTOR if an immediate reply is required) and wherever possible, the use of EXCP or SVC0 for issuing messages should be avoided. Action messages issued via WTO are deleted from the console via the DOM macro when the operator has performed as indicated, using the message number supplied by the WTO macro, except, for example, CICS and PSF action messages delivered via the DDS interface (message 1QZ2A) which have to be manually deleted. The utility IPW$$DD will not use WTO/WTOR/DOM macros.

Using the WTO or WTOR macro, each message indicates:

- a "routing code"
  indicating the console(s) to receive a copy of the message (see Figure 55 on page 145).
- a "descriptor code"
  indicating the type of message with its corresponding color (see Figure 56 on page 146).

For an explanation of the rules for coding message routing and descriptor codes and their defaults see “Message Reference” on page 401.

Messages issued via an EXCP or SVC0 macro receive default routing and descriptor codes assigned by VSE (routing code "Master Console Information" and descriptor code "System Programmer/Maintenance/Error").

```
 WTO Routing Code:                                Type: (IPW$GMD RT=)
 -----------------                                -----------
 1  = Master Console Action                           MA
 2  = Master Console Information                      MI
 3  = Tape Pool                                       TA
 4  = Direct Access Pool                              DI
(5  = Tape Library ) (*)
 6  = Disk Library                                    DK
 7  = Unit Record Pool                                UR
 8  = Teleprocessing control                          TP
 9  = System Secruity                                 SE
 10 = System Programmer/Error/Maintenance             SP
 11 = Programmer Information                          PG
(12 = Emulators)
(13-20 = reserved for customer use)
(21-28 = reserved for IBM/customer-defined subsystem use)
 32 = Hardcopy File only                              HC

where (*) = not used since POWER does not issue tape mount messages
            with a volume ID
```

Figure 55. Local Message Routing Codes for WTO/WTOR Macro

```
                                                       VSE/POWER Interpretation:
                                           VSE/POWER: <Cmd> <-Action--> <-Info-->
 WTO Descriptor Code:                Colour    Type:   | Cmd   Deci- Act- Sys. Info
 --------------------                          (IPW$GMD | Resp  sion  ion  Fail
                                                 DC=)  |
                                     -------   ------- | ----  ----  ---- ---- ----
 1  = System Failure                  Red  (H)   SF     |                   X
 2  = Immediate action required       White(H)   DA     |       X     X (a)
(3  = Eventual action required (*) )  Green             |
 4  = System status                     "        SS     |                    X (System)
 5  = Immediate command response        "        CM     |  X (c)
 6  = Job status                        "        JS     |                    X (Job)
(7  = Retain Action message for         "               |
       life-of-task         (**))                       |
 (8,9,10: not used by VSE)              "               |
 11 = Critical eventual action requested  Red  (H)   AK |             X (b)
 12 = Important information messages  Green      II     |                    X (Important)
 (13-16 = reserved)

where (*) = not used by VSE/POWER
     (**) = not used by VSE/POWER: if used then
            - any start-up errors highlighted in red would be almost immediately deleted
              from the screen (since VSE/POWER would immediately cancel the partiton)
            - any ABEND errors highlighted in red would be deleted as soon as the dump processing
              was finished
            - any other messages would remain on the screen anyway since VSE/POWER is a long
              running task
      (H) = Highlighted + hold on screen
      (a) = Action Messages to be Deleted by VSE/POWER
      (b) = Action Error Messages to be Deleted by Operator (e.g. I/O Error)
      (c) = Code is set automatically by IPW$$MS if handling AR Command.
            Cannot be set via IPW$GMD.
```

Figure 56. Local Message Descriptor Codes for WTO/WTOR Macro

Console Response Messages. If the message is a VSE Attention Routine commands response message, then additionally the message will be issued with:

- the console ID
- a correlation token

furnished by the VSE Attention Routine,

- and an indicator that the message is to be "connected" to other messages, causing the response messages to be buffered together for improved readability.

Exceptions to "connected" message handling:

- system error messages (e.g. 1QB5I INTERNAL MACRO CALL FAILED) will cause the "connected" message display to be interrupted and displayed immediately
- command response messages issued via the STXIT OC exit routine will not issue "connected" messages

Console response messages are routed by VSE to the origin console. An additional routing code is added by VSE/POWER to the Master console if needed.

Tagging Job Related Messages with Partition ID Messages which are issued during job execution will be displayed by VSE with a preceeding partition ID. This enables operator redisplay of partition-specific messages including VSE/POWER messages. Decision messages (e.g. 1Q55A SPECIFY TAPE

ADDRESS) will however be displayed with the VSE/POWER partition ID since such messages cause VSE/POWER to hang until the reply occurs.***:*** The field CAAPID in the CAT will be set by the task dispatcher with the partition ID of the execution task, and will be read by VSE when the message descriptor code is "Job Status" causing the message partition ID to be displayed.

Remote Message Request The function to be performed is indicated in the function indicator byte in the Remote Message Control Block. The following functions are performed:

- Queue remote messages (BSC and SNA) to the remote message queue.
  If the message is in NMR format, the originating node and/or remote/user id are put in front of the actual message. The message is truncated to its maximum length if applicable. Message modification is performed, if applicable by executing the IPW$GMS TYPE=SUB macro instruction. Then the message text is examined and multiple blanks are deleted from the text. Finally the message is anchored by means of a message index to the line control block if BSC or logical unit control block if SNA respectively.
- Delete messages from the queue when it is completely full with pending messages.
  When the remote message queue is full (255 entries) with pending messages, it is assumed that somebody is monopolizing the queue. This can be the case when a remote printer has not been ready for a while. All messages for that remote user are deleted and replaced by message 1R20I.
- Display ALLUSER-type messages by passing them to the command processor.
- Delete ALLUSER-type messages.
- Queue ALLUSER-type messages to the ALLUSER-type message queue. The ALLUSER type message queue contains only a limited number of entries (15). When the queue is full, the queue request is rejected.
- Delete BSC messages from the LCB subchain.
- Locate the first pending message for a specific BSC or SNA user.
- Delete SNA messages from the SNA delete subchain.
- Delete SNA messages temporarily by moving the entries from the SNA live subchain to the SNA delete subchain.
- Add temporarily deleted SNA messages to the SNA live subchain.

Nodal Message Request The function to be performed is indicated in register 0. The following functions are performed:

- Add nodal message record to appropriate node control block.
  On entry, the routine acquires storage to hold the NMR to be queued. If no storage could be obtained, the routine returns to the caller with R1=4. Otherwise the message text is copied from the NMR to the just acquired storage area. If the NMR contains a message, originated from the local node, message modification is invoked by issuing the IPW$GMS TYPE=SUB macro instruction. The network definition table is scanned to find the prime and, if specified, the alternate route node name. The NCB chain is now scanned to check if a connection is established with one of these nodes. If so, the NMR is queued at the tail of the message queue on this NCB. The network driver is then posted, to attach a console transmitter task, if one does not already exist. If the prime or alternate routing node is connected to another system, sharing the same queue file, the slot manager is called to pass the NMR across the shared spool by means of the IPW$IQS REQ=BUILDSLOT macro instruction.
- Build nodal message record and add it to appropriate node control block.
  On entry, the routine locates the message to be sent in the message definition module and acquires storage to hold the message. If no storage could be obtained, the message is discarded. Otherwise the message text is copied into the just obtained storage area thereby formatting a NMR. The destination node name and remote/user id are set up according to the caller's specification. The local node

name is inserted as originating node. Next, the routine branches to the add function to perform message modification and finally to queue the message to the appropriate NCB.

Message Distribution - IPW$GMS: The message distribution routine, which is part of the IPW$$MX module is invoked by means of the IPW$GMS TYPE=DIST macro instruction. The routine is responsible to distribute a message or command which is already in the NMR format or a message which is passed in internal format.
Depending on the destination, the routine directs the messages to:

- Local operator
- Any remote operator locally attached
- Any local VSE/ICCF user
- Any user on another node
- Any subsystem running on the local system, assuming a 'notify' communication path is established to the subsystem
- Any fixed format message queue

Commands are either passed to the 'invoke command processor' routine or forwarded to the next node on its way to the final destination.

Note: If the command is destined for the local system but in global command format, the command is discarded and message 1Q5FI is sent back to the originator.

If the message is destined for another system, sharing the same queue file, the slot manager is called to pass the NMR across the shared spool by means of the IPW$IQS REQ=BUILDSLOT macro instruction.

Message Distribution - SAS Local Message: Local messages generated by a spool-access support user task will be routed to that user except for:

- the 1R88I OK message, or
- special messages which instead are routed to the central operator (see IPW$$MS, table MM38 shown below):

```
*              THE FOLLOWING MESSAGE ID TABLE LISTS ALL MESSAGES
*              WHICH ARE NOT PASSED TO THE SAS USER.
MM38     DC    C'1Q38A '           Q-FILE DASD SOS MESSAGE     @DA15044
         DC    C'1QB8I '           RECOVERY COMPLETED INDICAT. @DY42689
         DC    C'1QF8I '           N FREE DBLKGP'S LOST        @DY42689
         DC    C'1Q61I '           IRREC. I/O ERROR ON D-FILE  @DY42689
MM32     DC    C'1Q32A '           A-FILE DASD SOS MESSAGE     @KD40385
MM31     DC    C'1Q31I '           A-FILE 80% MESSAGE          @DA15044
MM59     DC    C'1Q59I '           WAITING FOR REAL STORAGE    @D22BDWS
MM85     DC    C'1Q85I '           WAITING FOR VIRTUAL STORAGE @D22BDWS
MMX1     DC    C'1QX1I '           XPCC FUNCTION ERROR         @D22BDWS
MMX3     DC    C'1QX3I '           TASK STOPPED MESSAGE        @D22BDWS
         DC    C'1Q75I '           MULTIPLE TERMINATION OF TASK@D22BDWS
         DC    C'1Q76I '           VSE/POWER CAN NOT CONTINUE  @D22BDWS
         DC    C'1QB5I '           INT MARO CALL FAILURE       @D23BDWS
MMF0     DC    C'1QF0I '           SPOOL FULL PERCENTAGE       @D23BDWS
MMF4     DC    C'1QF4I '           NO FREE QUEUE RECORD AVAIL  @D23BDWS
         DC    C'1QZ0I '           SEVERE INTERNAL ERROR       @DA38824
         DC    C'1Q6GA '           INVALID DBLK NO (I/O-ERROR) @DA41998
         DC    C'1Q6HA '           DBLK WITH SER, BUT WITHOUT  @DA41998
*                                  LAST DBLK FLAG  (I/O-ERROR) @DA41998
         DC    C'1Q6JI '           I/O-ERROR DURING READ       @DA41998
         DC    C'1Q6KA '           DBLK WITHOUT SER, BUT WITH  @DA41998
*                                  LAST DBLK FLAG  (I/O-ERROR) @DA41998
         DC    C'1Q6LA '           RECORD LENGTH=0 (I/O-ERROR) @DA41998
         DC    C'1QBAI '           QUEUE FILE RECOVERY         @DA41998
         DC    C'1QFAA '           FREE DBLK IS 'IN USE'       @DA41998
         DC    C'1QFBA '           DBLK FREED IS ALREADY FREE  @DA41998
         DC    C'1QFCA '           DBLK NUMBERS MISMATCH       @DA41998
         DC    C'1QFDA '           FREE DBLK NUMBERS MISMATCH  @DA41998
         DC    C'1Q6UA '           DBLK GROUP OWNEP MISMATCH   @D67QDAT
         DC    C'1Q6VA '           DISPLAY 1Q6UA SEH RECORD    @D67QDAT
```

Message Blank Compression: The message blank compression routine, which is part of the IPW$$MX module is invoked by means of the IPW$GMS TYPE=SQUEEZE macro instruction.
The message text is scanned for duplicate blanks. If two or more consecutive blanks are found, the blank characters are removed from the message text and the message length is adjusted accordingly. The following table (table NOCOMPTB in IPW$$MX) lists messages which are not compressed:

```
NOCOMPTB DS    0H                  NOCOMPRESS. MSG.-TABLE      @D34BDSN
         DC    C'1R46'             MESSAGE-ID                  @D34BDSN
         DC    C'1Q40'             MESSAGE-ID                  @D34BDSN
         DC    C'1QB9'             MESSAGE ID '1QB9'           @D22DDWS
         DC    C'1R41'             MESSAGE ID '1R41'           @D23IDSW
         DC    C'1R47'             MESSAGE ID '1R47'           @D35YI38
         DC    C'1R48'             MESSAGE ID '1R48'           @D35YI38
         DC    C'1RB7'             MESSAGE ID '1RB7'           @D03PIPH
         DC    C'1Q6A'             MESSAGE ID '1Q6A'           @D51MDAT
         DC    C'1R4A'             MESSAGE ID '1R4A' (D EXIT)  @KX41033
         DC    C'1R4B'             MESSAGE ID '1R4B' D CRE‘DEL @D65EDMW
         DC    C'1RTF'             DISPLAYS TCP/IP DATA        @D65CDHS
```

Message Modification: The message modification routine, which is part of the IPW$$MX module is invoked by means of the IPW$GMS TYPE=SUB macro instruction.
The message text is scanned for a message modifier character and if found the appropriate modification is done. See Figure 57.

Mod ID	Message Modification	Obtained from where
X'01'	PNET BSC trace input	NCB and Reg. 15
X'02'	PNET BSC trace output	NCB and Reg. 15
X'03'	PNET SNA input buffer information	NCB and Reg. 15
X'41'	BSC transmission count	NCB / LCB
X'42'	BSC time out count	NCB / LCB
X'43'	BSC error count	NCB / LCB
X'44'	Node name	Node control block
X'62'	Member name.member type	SL member element
X'63'	Macro name (librarian)	Register 15
X'64'	Return code/feedback code (librarian)	Register 15
X'71'	Number of PNET SNA sends	Node control block
X'72'	Number of PNET SNA receives	Node control block
X'73'	From node and user id	Queue record
X'74'	First operand	Command control block
X'75'	Original job number	Queue record
X'76'	From node / user	Command control block
X'77'	First 60 bytes of operands	Command control block
X'78'	Command operand number	Command TCB control
X'80'	8–byte field	Register 4
X'81'	Sense information	VTAM RPL
X'82'	CCB address	Register 7
X'83'	Unit address	Task control block
X'84'	RJE identifier	Line control block
X'85'	RTNCD,FDB2	VTAM RPL
X'86'	Forms id	Queue record
X'87'	Return code, reason code	Register 15
X'88'	3800 printer setup message	Queue record
X'89'	Application id	VTAM ACB
X'8A'	Tape address	Tape control block
X'8B'	UCS phase name	TCB extension area
X'8C'	Keyword in error	Register 7
X'8D'	File name	Register 4
X'90'	Task ID bytes 2+3	TCB TCTI
X'91'	Job name	Queue record
X'92'	Target node name	Queue record
X'93'	RJE,BSC line address	Line control block
X'94'	LU name	LUCB
X'95'	Job number	Queue record
X'96'	Command code	Command control block
X'97'	SNA stop code	WACB
X'98'	RPL request	VTAM RPL
X'99'	RJE identifier	Task control block
X'9A'	5–digit number	Register 4

Figure 57 (Part 1 of 4). Message Modification Characters and Action Table

Mod ID	Message Modification	Obtained from where
X'A2'	Local SYSID	DMB
X'A3'	Task identifier	Task control block
X'A4'	User information	Queue record
X'A5'	RJE identifier	Register 0
X'A6'	BIND data	Register 15
X'A7'	3-digit number	Register 4
X'A8'	Logon reason code	Register 15
X'A9'	JOB/OUT constant	Queue record
X'AA'	Current date	COMREG
X'AB'	Dynamic Class Table Identifier	Register 14, byte 3
X'BA'	Current time	Via IPW$RDC macro
X'BB'	NJE line address	Node control block
X'BC'	NJE node name	Node control block
X'FF'	Second level modification id	–
	Second Level Modifiers	
X'00'	Diskette device address	3540 PWS
X'01'	Job return code	Part. control block
X'02'	Reason code	Register 7
X'03'	Phase name	Register 14
X'04'	XPCC function code	XPCCB
X'05'	XPCC return code	XPCCB
X'06'	External device name	EDCB
X'07'	DDS name	EDCB
X'08'	DDS name	EDCB, addressed by R8
X'09'	Command originator	EDCB
X'0A'	Command verb	EDCB
X'0B'	Spool access support task token	SAS work area
X'0C'	XPCC application & user id	XPCCB
X'0D'	8-bytes character string	Register 14
X'0E'	Queue file full percentage	DMB
X'0F'	Data file full percentage	DMB
X'10'	12-digit number	Register 4
X'11'	12-bytes MCB storage descriptor	Register 4
X'12'	Queue type (RDR/LST/PUN/XMT)	Queue record
X'13'	12-digit number	Register 7
X'14'	2-byte reason code	Task control block
X'15'	Lost DBLK group percentage	Register 4
X'16'	CKD/FBA disk address	MCB, addressed by R4
X'17'	DBLK size	DMB
X'18'	DBLK/Queue record block number	MCB, addressed by R4
X'19'	Account file cuu	ACB, addressed by R4
X'1A'	DBLK group size	DMB
X'1B'	Job suffix number	Queue record
X'1C'	Insert character in quotes in msg text	rightmost byte R15
X'1D'	Insert 4 printable chars in msg text	Register 15
X'1E'	Insert 2 printable chars in msg text	Register 15
X'1F'	DBLK size from generation table	Generation table
X'20'	cuu of execution writer task	TCB for new task

Figure 57 (Part 2 of 4). Message Modification Characters and Action Table

Mod ID	Message Modification	Obtained from where
	Second Level Modifiers (cont.)	
X'21'	Insert 5 decimal digits	Register 7
X'22'	Insert 8 hex digit address	Register 15
X'23'	Insert 23 bytes of Trace table ID	Trace Table address
X'24'	Insert 7 byte SLI Library Name	SLDS
X'25'	Insert 50 byte SLI Lib.+Sublib.names	SLDS
X'26'	Insert 8 byte local cpu SECNODE name	Trace Table address
X'27'	Insert 5 byte Trace Module Ver/Mod lvl.	CATC pointer
X'28'	Insert 8 byte Tracing buffering method	CATCCB pointer
X'29'	Insert "LOG=NO" if specified in qrec	TCQV
X'2A'	Insert VSE Security Userid	PCE Security Token
X'2B'	Insert 8 byte Tracing buffer type	CATCCB pointer
X'2C'	Insert application-id and user-id	ACIE via R5
X'2D'	Insert 'DUE TO EXIT FAILURE'	TCF11 TCF15
X'2E'	Insert 'BY OPERATOR' or see X'2D'	TCF15
X'2F'	Insert I/O CCB for 1R30I	Register 7
X'30'	Insert I/O CCW for 1R30I	Register 8
X'31'	Insert Hex contents of Register 7	Register 7
X'32'	Insert 8 bytes Console Name (CPCON)	TCB
X'33'	Insert Hex contents of Register 8	Register 8
X'34'	Insert userid	QRTU
X'35'	Show 1 byte as bits	Reg.15 (left byte)
X'36'	Inserts device code from QRDT	QRDT
X'37'	Inserts device code from TCDT	TCDT
X'38'	Inserts 'LUNAME=' if SNA task	n/a
X'39'	Inserts LTA phase name	XRWDS.XRWLTA
X'3A'	Inserts tape volume number or '***'	QRDS.QRVOL
X'3B'	Inserts 3 digits	TBDS.TB1QG01
X'3C'	Inserts 3 digits	TBDS.TB1QG02
X'3D'	Inserts 3 half-bytes	Reg.15 right 2 bytes
X'3E'	Inserts queue id (RDR,LST,PUN,XMT)	(calcu'd from q-ptr)
X'3F'	Inserts queue class + x'40'	(calcu'd from c-ptr)
X'40'	Inserts hex character	Reg.15 left byte
X'41'	Insert Task Id found in TIK in XECBTAB	Reg.15
X'42'	Insert CUU of queue record field QRCU	TCQV -> QRDS
X'43'	Inserts upto 85 characters, length is found at 3rd position (FF43LL)	Reg.5 -> char. string
X'44'	Insert original I/O CCW for 1R30I	XRWCWDA & XRWCWCT
X'45'	Insert 32 hex characters in blocks of 8 characters each	Reg.4 -> data Reg.0 = length
X'46'	Complete 1R56I8/9 with NCB related inf.	TCENCB
X'47'	Insert In/Outbound RUSizes	LUCB (LUBSI/LUBSO)
X'48'	Number of DFILE Extents	Register 14
X'49'	DFILE Extent start,length	Area pntr register 15
X'4A'	Substitute 'SSL' for 'IP' in 1RTx	Flag MMMF1SSL
X'4B'	5 Digit number	Register 14
X'4C'	9 Digit number appended by - Track/Block information in MCB	Reginser 4 Reginser 6
X'4D'	11 Digit number, then bumps reg.4 by 4	Register 4

Figure 57 (Part 3 of 4). Message Modification Characters and Action Table

Mod ID	Message Modification	Obtained from where
	Second Level Modifiers (cont.)	
X'4E'	32 SEH bytes, returns in register 15 - the number of bytes/4	Register 7
X'4F'	8 Bytes Jobname JCA	JCAJOBNM
X'50'	Number of formatted Data File Exten's	Register 4
X'51'	Offload Type	TCOCMDT
X'52'	1Q5MI TRACE= data	JCA trace bytes: JCATROF,JCATROFS, JCATRPS+4,JCATRTRO+6
X'53'	Hex TCQW Queue Record number	TCQW
X'54'	3 digit number in register 14	Register 14

Figure 57 (Part 4 of 4). Message Modification Characters and Action Table

Message Coding and Documentation Considerations: For rules concerning the coding of local messages see “Message Reference” on page 401 for more details.

The following definitions are to be used when documenting messages in the Message Manual:

(Central) Operator For VSE/POWER, any operator other than a User Programmer, System Programmer or Remote RJE Operator.

Operator Response Indicates action to take for:

- console/hardware operation
- hardware error correcrtion
- hardware error circumvention
- "Notify System Programmer" if:
  - problem system performance
  - problem system generation
  - problem hardware if circumvention needed
  - changes in system generation or layout
  - security problem
  - indicated in Message Manual
- "Notify Programmer" if:
  - offline job problem/error (eg reading job into POWER RDR queue or LST/PUN task problems) since such errors or messages are not sent to a user terminal or included in a job console listing.

System Programmer Response Indicates action to take for:

- system software error (generation and/or logic error)
- system generation and/or layout changes
- hardware problems causing performance degration
- aid with problem circumvention

Programmer Response Indicates action to take for:

- specific job execution problems
  - sortware and JCL
  - hardware (diskettes, tapes or other machine readable job input/output)
- Specific job offline problems (RDR task, LST/PUN task, POFFLOAD error)

After VSE/POWER 5.2 the developer and change team will have to consider for each new locally displayed message in the future:

- The message routing code (see Figure 55):
  - the message routing is determined by:
    1. routing code (explicit or default)
    2. console ID (VSE/POWER AR commands)
    3. console name (not used by VSE/POWER for command message routing - instead used for security checking)
  - messages issued via EXCP/SVC 0 for VSE/POWER will have default routing codes of 2 (Master Console Information) and 10 (System Programmer/Mainenance/Error). This includes the IPW$$DD utility.
- The message descriptor code (see Figure 56):
  (colour, highlighting and hold on screen or not):
  - message descriptor determined by descriptor code (explicit or default)
  - three different message colors are available for VSE depending on the descriptor code (vendors or customers may use different rules):
    - red (code=1 or 11), highlighted and held on screen,
    - white (code=2), highlighted and held on screen,
    - and green (all other codes).
  - some messages issued via EXCP for VSE/POWER will have a descriptor of 1 (System Failure: red colour highlighted). These messages are contained in a VSE exception list (see “VSE "Exception List" Messages for VSE/POWER” on page 407)
  - All decision (i.e. reply) messages are set to white highlighted by VSE. This includes the IPW$$DD utility.
- Any message normally issued by EXCP that can't use the default routing and descriptor codes will have to be issued by WTO/WTOR or be included in the VSE Message Exception List (see “VSE "Exception List" Messages for VSE/POWER” on page 407)
- A command message response is different from other messages. Its routing will be to the console that issued the command, and additionally to any other routing specified (e.g. system error messages that should be also routed to the System console). This can only be done using the WTO/WTOR macros (the AR Appendage Interface passes the Console ID and a correlation token (CART) for properly returning the message response).
- The saving of a message ID for later deletion by DOM macro:
  - any message issued to be deleted automatically from the screen later will have to issue the WTO macro which returns a message identifier to be saved for reference by a DOM macro to delete the message later when the appropriate action is taken (decision messages are deleted automatically when replied).
  - the appropriate routine(s) will have to be chosen to implement and issue the DOM macro (command e.g. PGO,PACCOUNT,PSTOP; task termination)
  - task termination code may have to cleanup an action message by issuing the DOM if an error condition occurs (e.g. disk I/O error).
- All highlighted messages (e.g. red system failure messages) should be deleted via the DOM macro if possible, for example, when a hardware error occurs and is later corrected. This deletion will save VSE system storage and save the operator the manual work. Note the system will not fail since VSE will begin deleting such messages when too many occur.

  Note: Some highlighted messages are not meant to be deleted by the system, rather they are to be deleted by the operator (see Descriptor Code "AK", e.g. as with the messages 1Q6GA, 1Q6LA)

  Note: Decision messages issued by WTOR are always highlighted and automatically deleted after the operator reply.

- The use of chained multi-line "connected" message techinques for displays which are logically related (e.g. status displays) must:
  - Use the WTO CONNECT= parameter
  - Requires saving the 1st message ID from the WTO macro and using it a s input to subsequent WTO CONNECT=(message id) macros.
  - Maximum message length is 70 bytes (WTO specs - for VSE/POWER we use 69 to conform to old hardware specifications). Long messages will be split into two lines of 69 bytes each and the second line will be right adjusted.
  - Non-last message has linetype=D
  - Ending message has linetype=DE or use a dummy message with linetype=E.
  - **Note** - the last connected message **MUST** be issued to cause the VSE message buffer to be emptied and sent to the console (see implementation hits in task termination IPW$$TR and VSE/POWER ABEND termination IPW$$AT).
- WTO/WTOR implementation details:
  - The WTO does **NOT** post an ECB when a message is processed as a SVC 0/EXCP would do, i.e. if the WTO goes into a "busywait" on storage then the code can't simply call a VSE/POWER wait macro IPW$WFx and have the task wait on posting - the task has to use nucleus timer wait facilities
  - The maximum single line is 125 bytes for the WTO macro (not POWER's 132 bytes) and decision messages requiring an immediate reply have maximum 122 bytes for the WTOR macro. A WTO messages greater than 125 bytes will be split into two lines of 69 bytes and left adjusted, unless it is a console connected display message in which case it will be right adjusted.
  - The message length for the WTO/WTOR macros is two bytes
  - Job-related messages issued from some VSE/POWER task for some spooling partition are to be "tagged" with the partiton ID (Note: VSE support for this function is only for the VSE/POWER maintask).

Normal Message Handling: The following implementation was adopted for locally issued messages using IPW$GAM or IPW$GAM+IPW$WTO:

1. Message definition (in macro IPW$GMM using the macro IPW$GMD):
   the developer will need to specify the routing and descriptor code(s) for the individual message if the defaults are not sufficient:

   ```
   IPW$GMD ...,RT=(route-code1,rout-code2,...),DC=desc-code
   ```

2. Issuing a single message to the operator will then require no further action if done by:
   - the macro

     ```
     IPW$GAM MSG=...,DEST=LOCAL   (normal case)
     ```

   - Combinations of the following IPW$GAM to obtain a message:

     ```
     IPW$GAM MSG=...,DEST=RETURN  (return message address in Reg.3)
     IPW$GAM MSG=...,DEST=address|(Rx) (copy msg to buffer/storage)
     IPW$GAM MSG=...,REQ=ADDR     (copy msg addr to reg.1)
     ```

   with combinations of the following to issue the message (and obtain reply):

   ```
   IPW$WTO
   IPW$WTR
   ```

   For example, the Command Processor:

   ```
   IPW$GAM MSG=...,DEST=MESSOUT (copy msg to workarea)
   ```

   followed by the call of the subroutine:

```
         L    RF,MSG
         BAL  RE,RF      (issue IPW$WTO)
```

The IPW$GAM macro (DEST=RETURN/address/(Reg) or REQ=ADDR) will cause the routing and descriptor codes to be placed in the TCB (TCMRT and TCMDC) which are then used as input to the IPW$$MS (CAMS+24) for issuing the message (the codes are then cleared).

Messages issued as above will receive the full WTO/WTOR support (e.g. the routing and descriptor Code usage specified with the message in the Message Module IPW$$MM) and the developer will only need to specify the message routing and descriptor codes in IPW$GMM. Futher default handling of routing and descriptor codes occurs during issuing of the message in IPW$$MS.

Also the TCB will contain the message ID (TCMID) returned from the WTO macro which can be used later for message deletion (DOM) if necessary (e.g. when the PGO command is issued).

3. PNET NMR and Shared QCA messages arriving for the system console:
   - PNET XMIT command response messages (e.g. PDISPLAY)
   - PBRDCST messages
   - Job Notify messages

   do not issue the IPW$GAM macro in the format as directly above and therefore the message arrives at IPW$$MS (CAMS+24) (if the local CPU is the target system) with the TCB routing and Desciptor code fields containing zeros and will receive internal default handling:
   - default routing and descriptor codes
     (routing=Master/System Console and
     descriptor=System Status: green not highlighted)
   - no Connected message displays will be performed
   - any Action message will have to be deleted manually

Messages Requiring Special Handling

1. Messages issued by IPWWTO/IPWWTR not fetched by IPW$GAM, i.e. not by:

   ```
   IPW$GAM MSG=...,REQ=ADDR     (copy msg addr to reg.1)
   IPW$GAM MSG=...,DEST=RETURN  (return message address in Reg.3)
   IPW$GAM MSG=...,DEST=address|(Rx) (copy msg to buffer/storage)
   ```

   must store the routing and descriptor Codes to the TCB (TCMRT and TCMDC) prior to issuing the message (if defaults not acceptable).

2. Messages issued by IPWWTO/IPWWTR after calling the IPW$GAM DEST=RETURN/address/(Reg) or REQ=ADDR for some message A, and later the module issues a message B locally defined in the module will have to reset the TCMRT and TCMDC fields which will contain the routing and descriptor codes of message A.

3. Any new message being issued by EXCP (e.g. in IPW$$AT) will have to consider whether the default EXCP routing and descritor codes are proper, and if not, then instead the message will have to be issued by a WTO/WTOR macros or be included in the VSE Message Exception List (see “VSE "Exception List" Messages for VSE/POWER” on page 407).

4. Command Processor considerations (IPW$$Cx):

   - Any message issued by the Command Processor should use the central IPW$$CM subroutine MSG which automatically processes Connected message without the caller being aware of it (otherwise the message would not appear in the Connected message display).

   - Any decision message issued by the Command Processor should set the flag SWFLAG2.SWREPLY=on to cause the IPW$WTR macro to be issued.

   - Any task attached by the Command Processor to issue command response messages (e.g. similar to IPW$$PS) requires that the response Console ID and CART are passed to the new task

- If a message requires message substitution using registers 14 or 15 should use the macro:

  ```
          IPW$GAM MSG=...,DEST=MESSOUT,SUB=YES
  ```

  instead of DEST=LOCAL which destroys those registers before performing message substitution.

- Error messages issued by the Command Processor (i.e. 1QB5I, 1QZ0I) should be issued directly to the system console instead of the the origin console:

  ```
          IPW$GAM MSG=$1QB5I,DEST=LOCAL
  or
          IPW$GAM MSG=$1QZ0I,DEST=LOCAL
  ```

  Note: This is important, because by issuing the message with DEST=LOCAL causes a possible connected message display to be interrupted by IPW$$MS, and the message display buffer prior this message to be immediately displayed, followed by this error message, so that the operator can see all messages preceeding the error condition.

5. Print Status considerations (IPW$$PS):

   - Any status message issued should use the central subroutine PSMSG which automatically processes connected message without the caller being aware of it.
   - Any normal message issued to the local central operator should use the central subroutine PSMSGLOC which automatically processes connected messages.
   - Error messages issued by the Command Processor (i.e. 1QB5I, 1QZ0I) should be issued directly to the system console instead of the the origin console:

     ```
             IPW$GAM MSG=$1QB5I,DEST=LOCAL
     or
             IPW$GAM MSG=$1QZ0I,DEST=LOCAL
     ```

     Note: This is important, because by issuing the message with DEST=LOCAL causes a possible connected message display to be interrupted by IPW$$MS, and the message display buffer prior this message to be immediately displayed, followed by this error message, so that the operator can see all messages preceeding the error condition.

6. Action message considerations.

   Action messages are of types (a) and (b), where by the former has the Descriptor Code "DA" and the latter Code "AK". For the former, the message ID must be saved for later DOM'ing, and care taken to delete the message only when appropriate (e.g. when an incorrect reply is made to an action message, then the message should not be deleted). The latter require manual deletion by the operator.

Notify Processing

The Notify message routine is located in the IPW$$NS module. Whenever an IPW$NTY macro is issued and the message is destined for

- a local VSE/ICCF user
- VSE/DSNX
- a fixed format message queue
- any subsystem running on the local system

The Notify message routine is called by the Notify service routine in the VSE/POWER nucleus.

Depending on the recipient, the following functions are performed:

- Message destined for VSE/ICCF user:

If the message is not already in NMR format, virtual storage is acquired to build an NMR and the message is converted into that format. If the message originated at the local VSE/POWER system, an IPW$GMS TYPE=SUB macro instruction is issued to perform message modification. The message is then queued at the tail of VSE/ICCF Notify message queue and the notify task is posted.

If the maximum number of messages to be held in core is exceeded, the oldest message (first message in queue) is removed from the queue and discarded. If currently no virtual storage is available to hold the message, the message is discarded. In both cases the operator is informed via message 1RA2I and the lost message count is updated for statistical purposes.

- Message destined for VSE/DSNX:

  If the message is the job completion message 1Q5DI, virtual storage is acquired for the fixed format message record and the message is then converted into that format. The fixed format message record is then added at the end of the VSE/DSNX message queue and the notify task is posted to forward the message to VSE/DSNX. All other type of messages, however, are discarded.

- Message destined for subsystem:

  The communicator information element (CIE) chain is scanned to determine if a 'notify' communication path exists to the target subsystem. If not, the message is discarded. Otherwise virtual storage is acquired to build an internal message record, if the message is not already in that format. The message modification routine is invoked by means of the IPW$GMS TYPE=SUB macro instruction. The message is then queued at the tail of the appropriate subsystem message queue. Finally, the notify task is posted to forward the message to the appropriate subsystem.

- Message destined for a fixed format message queue:

  This is applicable only for messages in fixed format presentation. The message is converted from internal format to nodal message record format. The message data within the nodal message record is then easily accessible by application programs via DSECT JCMDS provided by the PWRSPL macro. The message can be sent to another node, another system, or queued locally to a fixed format message queue. The message is always routed back to the node and system where the job originated. Such messages are created by the IPW$NTY QCM=YES macro instruction, if the job was submitted via VSE/POWER's Spool-access support with the 'queue completion message' option of the SPL.

Notify Task: The notify task is attached by the spool access service master task when the first 'notify' communication path is established between VSE/POWER and a subsystem, like VSE/ICCF.
The VSE/AF communicator support is utilized for the exchange of messages. The message transfer is one directional (e. g. VSE/POWER -- VSE/ICCF). Basically the notify task waits for the arrival of messages and forwards the message then to the desired subsystem, such as VSE/ICCF. If the notify task is posted with the indication that a message is queued for a particular subsystem, the notify task checks if message transfer is currently in progress for the subsystem the message is destined to. If so, the routine waits until the message transfer is completed. Otherwise the head message is unchained from the subsystem message queue and sent to the counterpart. Once the message is successfully sent, the storage occupied by the message is freed. This process continues until all messages are sent.
If, meanwhile, the connection to the subsystem breaks due to normal/abnormal termination, the notify task prepares itself for a new 'notify' connection with the subsystem.

If a severe error is encountered while sending messages to the counterpart, the system operator is informed via messages 1QX1I and 1Q4BI, and the 'notify' connection to the subsystem concerned is terminated. The notify task continues to service other 'notify' connections and waits for a new connection request from the subsystem facing the error.

The notify task is terminated by the spool access service master task at VSE/POWER termination time when no other tasks than the minimum tasks are active.

Asynchronous Service

The asynchronous service function of VSE/POWER handles all of the following requests:

- SETPRT
- LFCB
- OPEN/EOV/CLOSE
- LOAD
- OBTAIN SVA ENTRY POINT
- Dump particular storage areas
- Communicate with the Librarian

This is done for the following reasons:

1. Most of the called VSE/AF service routines run in the SVA under the TIK of the VSE/POWER main task. Any page fault which occurs cannot be correctly handled by the VSE/POWER page fault overlap processing, because the register convention used by the VSE/AF service routines do not match the VSE/POWER requirements.
2. Any I/O done by the VSE/AF service routines would cause the complete VSE/POWER partition to be put into the wait state until the I/O completed.
3. Any SVC-call would cause the VSE/POWER main task to wait until the service request completed.

The asynchronous service module consists of the following sections:

- ATTACH appropriate subtask
  - Asynchronous service subtask
  - Dump subtask
  - Librarian subtask
- DETACH appropriate subtask
  - Asynchronous service subtask
  - Dump subtask
  - Librarian subtask
- INVOKE appropriate subtask
  - Asynchronous service subtask
  - Dump subtask
  - Librarian subtask
- Asynchronous service subtask
- Dump subtask
- Librarian subtask

The Asynchronous service subtask is attached whenever a VSE/POWER task issues an asynchronous service request and detached when the request is processed. The Dump subtask is attached at VSE/POWER initialization time when the corresponding UPSI bit, requesting trace logging, is set or when the PSTART DUMPTR command is given and detached at VSE/POWER termination time. The Librarian subtask is attached at VSE/POWER initialization time when SUBLIB= or MEMTYPE= is specified in the VSE/POWER generation macro and is detached at VSE/POWER termination time.

Invocation of Asynchronous Service: Linkage to the asynchronous service function is established by the IPW$IAS macro instruction.

When the function is entered the first time, storage for the asynchronous service anchor block is reserved. The ASAB exists as long as VSE/POWER is active. The asynchronous service anchor block contains:

- Pointers to first and last entry in the various service request queues. A separate queue exists for the DUMP and the Librarian subtask.
- Subtask communication fields.
- Address of SETPRT routine in SVA.
- Address of VSE/ICCF Librarian routine
- Lockword.

When a DUMP or Librarian request is issued by a VSE/POWER task, the service request block (pointed to by register 1) is chained as the last entry in the appropriate service request queue, and the corresponding subtask is posted. The asynchronous service anchor block (ASAB) is unlocked and the task waits for the completion of the service request. After the completion of the service request, its ECB is posted by the subtask. The return code set by the subtask is analyzed and the appropriate action is taken.

When an asynchronous service request is issued by a VSE/POWER task, the service function reserves virtual storage for an asynchronous service work element (ASWE) and attaches the asynchronous service subtask. The ASWE contains among others the PSW, the register save area and the abnormal save area for the subtask. If currently no subtask is available the VSE/POWER task will wait 5 seconds and try to attach the subtask again. After completion of the service request the ECB in the SRB is posted by the subtask. The ASWE is released and the subtask is detached. If the subtask terminated abnormally IPW$$AT gets control and indicates 'request failed' in the SRB before posting the request initiator.

The asynchronous service function is serially reusable and is locked for the duration of the appropriate function (ATTACH, DETACH, or SERVICE).

IDUMP in Flight Function

This function allows to request a formatted dump of the VSE/POWER partition and of important Supervisor areas to the dump (sub)library assigned to the VSE/POWER partition. The function is requested through the macro IPW$IDM, which may be placed into nearly all VSE/POWER modules. The request macro itself and the appearance of messages accompanying an IPW$IDM request are described in detail in VSE/POWER Administration and Operation, Appendix B.

Requesting macro IPW$IDM in any piece of VSE/POWER or user exit code should not destroy caller registers. Therefore the macro expansion distinguishes between three types of callers and saves initially some important registers -

1. for private VSE/POWER task in TCB location TCIE
2. for VSE Subtask of VSE/POWER in save area permanently addressed by register RC
3. for call by module IPW$$AT in local save area at AT0E

Then module IPW$$ID is called, which ensures by gating mechanisms that only one requestor of each type 1) - 3) is allowed to enter the IPW$$ID processing code in order to build an IDUMP symptom record in its predefined area, to request the IDUMP macro, to open the gates again, and finally to return to its caller.

Control of IPW$IDM Expansion: In order to control the expansion of this macro, the following predefined compile time globals are used:

&$MNSA

Refuses to expand the macro, but provides MNOTE instead, because the code area must either be protected against recursive entry or it does not adhere to the VSE/POWER register conventions.

&$ATSA

Uses local IPW$$AT save area, because all callers of this module are serialized through locking of 'ATGATE'.

&$STSA

Uses save area permanently addressed by register RC as established at Subtask startup or at VTAM-exit entry. The save area lies mainly in local module area, because the subtask code need not be re-entrant (apart from asynchr. service subtask in IPW$$AS)

&$BTSA

Provides execution time decision code to follow either VSE Subtask expansion (as &$STSA) or VSE/POWER private task expansion (as &$PWSA). Such provisions are necessary in module IPW$$SR and IPW$$MX, which may be driven both by the PNET SNA subtask and the line driver VSE/POWER task.

&$PWSA

Uses the VSE/POWER task TCB save area at TCIE. It represents the most common call by a VSE/POWER private task from a general VSE/POWER module. &$PWSA is defined and set implicitly by the IPW$DSD storage descriptor macro used in every VSE/POWER module.

&$NUSA

Operates as &$PWSA, and is used in IPW$$NU.

Meaning no global provided as typically in user exit code. The same save area is used as with &$PWSA, because exit code should be driven by VSE/POWER private tasks only.

Apart from extra expansion code for the macro operand DO= and FAIL=, the following instructions are generated when calling IPW$IDM:

1. Power task within VSE/POWER module

```
STM    RE,R1,TCIE        SAVE RE-R1 IN TCB
LA     R1,xySD           POINT TO MODULE STORAGE DESCRIPTOR
L      RF,CAID           GET ENTRY POINT IPW$$ID
BAL    RE,16(,RF)        ENTER MODULE IPW$$ID
LM     RE,R1,TCIE        RESTORE RE-R1 FROM TCB
```

   Power task within user exit routine

```
STM    RE,R1,TCIE        SAVE RE-R1 IN TCB
LA     R1,phasename-area POINT TO USER DEFINED NAME
ICM    R1,8,*+8          SET R1 HIGH ORDER X'80'
B      *+6               BYPASS LOCAL DEFINITION
DC     XL2'8000'         IDENTIFY CALL FROM EXIT ROUTINE
L      RF,CAID           GET ENTRY POINT IPW$$ID
BAL    RE,16(,RF)        ENTER MODULE IPW$$ID
LM     RE,R1,TCIE        RESTORE RE-R1 FROM TCB
```

2. Subtask (base always R9)

```
STM    RE,R2,0(RC)       SAVE RE-R2 IN RC->AREA
LA     R1,xySD           POINT TO MODULE STORAGE DESCRIPTOR
L      RF,CAID           GET ENTRY POINT IPW$$ID
BAL    RE,24(,RF)        ENTER MODULE IPW$$ID
LM     RE,R2,0(RC)       RESTORE RE-R2 FROM RC->AREA
                         NOTE: SUBTASK MODULE USED SERIALLY
```

3. $$AT (base R8, R9) call (with R2 returned for 2nd IDUMP request)

```
         STM    RE,R2,AT0E        SAVE RE-R2 IN LOCAL AREA
         L      R2,ATWASA         POINT TO FAILURE SAVE AREA
         L      R1,ATWNAME        POINT TO FAILING PHASE NAME
         LA     R0,ATWCCD         POINT TO CANCEL CODE
         L      RF,CAID           GET ENTRY POINT IPW$$ID
         BAL    RE,20(,RF)        ENTER MODULE IPW$$ID
         LM     RE,R1,AT0E        RESTORE RE-R1 FROM LOCAL AREA
*                                 WITH R2-->IDUMP PARM LIST
*                                 NOTE: $$AT CODE USED SERIALLY
```

4. Power task or Subtask in module driven by both types (IPW$$SR/MX, respecting Subtask convention to offer '$IDM' identifier at offset 24 in save area addressed by register RC)

```
         CLC    *+14(4),24(RC)    subtask identifier found ?
         BE     $STnnnn           ..yes, go and handle subtask
         B      $PTnnnn           go and handle power task
         DC     CL4'$IDM'         vse subtask identifier
$PTnnnn  DS     0H
         STM    RE,R1,TCIE        SAVE RE-R1 IN TCB
         LA     R1,xySD           POINT TO MODULE STORAGE DESCRIPTOR
         L      RF,CAID           GET ENTRY POINT IPW$$ID
         BAL    RE,16(,RF)        ENTER MODULE IPW$$ID
         LM     RE,R1,TCIE        RESTORE RE-R1 FROM TCB
         B      $CTnnnn           GO AND CONTINUE
$STnnnn  DS     0H
         STM    RE,R2,0(RC)       SAVE RE-R2 IN RC->AREA
         LA     R1,xySD           POINT TO MODULE STORAGE DESCRIPTOR
         L      RF,CAID           GET ENTRY POINT IPW$$ID
         BAL    RE,24(,RF)        ENTER MODULE IPW$$ID
         LM     RE,R2,0(RC)       RESTORE RE-R2 FROM RC->AREA
$CTnnnn  DS     0H
```

Operation of module IPW$$ID:

The header of module IPW$$ID provides a chart describing the logic structure of the IDUMP processor and the function flow as described below:

1. Entry point processing for a call by ...

   priv. VSE/POWER task (in NP or PA mode)

   set off Page Fault Handling Overlap (PHO) not to be interruptible by another VSE/POWER task during IPW$$ID processing and avoid any IPW$WFx call for the same reason.
   Request non-parallel (NP) mode by local TDSERV request as required for later IDUMP call.
   For details refer to “Multiprocessor Support” on page 111.
   Address the local save and symptom record area reserved for this type of caller.

   VSE Subtask (always in NP mode)

   LOCK IDGATE against concurrent usage by another VSE Subtask. Address the local save and symptom record area reserved for this type of caller.

   IPW$$AT module (always in NP mode)

   Respect that only one call at a time can come from IPW$$AT, because that module has already been gated by locking of ATGATE. Address the local save and symptom record area reserved for this type of caller.

2. Build symptom record for IDUMP to be written into the VSE/AF Dump (sub)library assigned to the VSE/POWER partition. VSE/POWER provides a symptom string, containing information about the cause of error, for problem determination. The symptom string is subdivided into 6 pre-defined sections, whereby only sections 1,2,3 and 6 are used by VSE/POWER

Section 1: This is the environment section; it describes hardware and operating system environment. The section is completed by VSE/AF.

Section 2: This section contains offsets to and length of the succeeding sections; the base for the offsets is byte 0 of the symptom string.

Section 3: This section contains error symptoms in structured data base format (SDB). This SDB format is the standard format used by the level one representatives for RETAIN searches. It contains the following VSE/POWER supplied information:

- Cancel code
- Component identifier (5686-033-01)
- Failing phase name, if available
- PSW/Registers difference

Section 6: This section identifies the major VSE/POWER control blocks:

- Control Address Table (CAT)
- Disk Management Block (DMB)
- Storage Control Block (SCB)
- Local Message Control Block (MMB)
- Virtual Storage Control Block (VSCB)
- Remote Message Control Block (MSCB)
- Dynamic Partition Control Block (DPCB)
- Communication Information Block (CIB)
- Communication Information Block 2 (CI2)
- Trace Information Block (TIB)
- Trace Facility Control Block (TRCB)
- Account Control Block (ACB)
- Module Control Blocks (MCB)
- Task dispatching trace if present
- PNET Master Control Block (PNCB)
- VTAM Driver Control Block for PNET (VDCB)
- RJE,SNA Master Control Block (SNCB)
- VSE/POWER Partition Control Blocks
- Task Control Block (TCB) chain
- Line Control Block (LCB) chain
- Node Control Block (NCB) chain
- External Device Control Block (EDCB) chain
- IPW$$NU with Control Address Table (CAT)

3. Request the IDUMP macro, which is an SVC 2, that must be called in NP mode. When the DUMP library is full or not defined, issue message 1QC5I and provide also an indication via the high order bit of register RB.

4. Return to caller processing for ...

priv. VSE/POWER task
Identify IPW$IDM request by message 1Q2JI, return to PA mode (unless task of NP-Must list), restore locally saved registers, switch on PHO again for private task concurrency, and return to expansion of IPW$IDM.

VSE Subtask
Identify IPW$IDM request by message 1Q2JI, restore locally saved registers, UNLOCK IDGATE for other subtasks to enter IPW$$ID, and return to expansion of IPW$IDM.

IPW$$AT module
restore locally saved registers and return to expansion of IPW$IDM (in module IPW$$AT).

Usage of IPW$IDM: The macro is currently used at about 70 existing code locations as summarized in the IPW$IDM reference of the IPW$$ID module header. Further APAR activity and new development should make use if IPW$IDM whenever

- any logic failure has to be captured by a VSE/POWER dump
- the failing task or function can be recovered or terminated smoothly.

Open/Close Tape

The module IPW$$OT performs several tape functions via the access macro IPW$OTP:

- construct tape control block
- delete the tape control block
- open tape
- close tape
- perform a BAM volume change for a continued queue record
- perform a BAM volume change for a non-continued queue record
- perform a BAM volume change for a SYSIN tape
- require the mounting of the last BAM tape volume for a given spool entry
- require the mounting of the first BAM tape volume for a given spool entry

This support is for non-accounting modules. The above functions are performed either for native tape support (labeled or non-labeled tape) and for BAM tape support (labeled or non-labeled tape). Native tape support is capable of reading labeled tapes only, but not writing labeled tapes. BAM tape support may read or write labeled tapes. The module IPW$$OT prolog has a detailed description of the functions and various tape formats.

Command Processor

The command processor (IPW$$CM) will be under control either of a permanent TCB located in the first page of the fixable area or a temporary TCB in the fixable area.

The permanent command processor task is invoked by the attention interface appendage when an operator command is received from the console.

The temporary command processor task is invoked by the IPW$ICP macro interface instruction.

On entry of the command processor the command to be analyzed and acted upon is contained in a command processor control block (CPB), which is part of the task control block (TCB). Once the command has beed identified, its phase entry address is obtained from the command processor table CMNDTAB located in IPW$$NU.

On exit, the temporary command processor task detaches itself and the permanent command processor task will place itself in inactive state. The permanent command processor has the highest priority of all common tasks in the task selection list. It enables the operator to maintain control over the VSE/POWER partition in extreme circumstances.

Initiation of the Permanent Command Processor Task

The attention routine will pass control to the attention interface appendage in IPW$$NU for a potential VSE/POWER command. In the appendage routine the command is verified and stored with its operands in fixed positions in the command processor control block (CPB). The command processor task is set dispatchable and normal return is taken.

In the case of an invalid command or if the command processor is already active an error return is taken, resulting in an invalid-statement message or a routine-active message, respectively.

If the attention routine is notified that the command processor is busy, then further commands will be rejected, and the attention routine will wait for posting by the command processsor when the initial command has been processed.

Initiation of the Temporary Command Processor Task

The VSE/POWER routine that wants to invoke the command processor for processing of a VSE/POWER command issues a IPW$ICP macro instruction. Processing control is then given to phase IPW$$IC.

This phase builds and attaches a temporary command processor TCB, with the command and its operands in fixed positions in its CPB. When called with the WAIT=NO option, IPW$$IC will return to the caller in case no real storage can be obtained for the temporary command processor TCB.
Commands received from another node are contained in the nodal message record (NMR). If a command is in NMR format, the originating node name and remote/user id are saved in the CPB. The network definition table is then scanned to check if the originating node is defined in the local system. If no entry is found, the command is thrown away. Otherwise command authority information are extracted from the network definition table entry of the node concerned.

Command Processor Organization

The VSE/POWER Command Processor is created by a single assembly of each subprocessor and main processor and a subsequent LINKEDT step. The main module (IPW$$CM), also called root phase, contains the entry point, the command formatting routine and all other subroutines.

At VSE/POWER initialization time all applicable command processor phases are loaded and their entry point addresses stored in the command table located in the command processor root phase.

Command Authorization Verification

Command authority check is made for each command either received from

- Local operator
- Remote operator
- CTLSPOOL user
- Another node, regardless of source
- Spool access service user (cross-partition user)

Any attempt to enter a VSE/POWER command without the correct authority results in message 1R85I or 1RA7I being issued and rejection of the command.

Command Processing Routines

In IPW$$CM, the command code contained in CPB is used to enter the appropriate command processing routine. Commands being entered from a connected remote station are first analyzed by the physical remote reader routine (IPW$$IB for SNA, or IPW$$BR for BSC) before it is passed to the command processor. There are some commands which do not affect the command processor task and, therefore, will be processed by the reader itself.

The following commands are processed by the IPW$$IB inbound processor:

- FLUSH
- GO
- RESTART
- SETUP
- SIGNOFF
- START
- STOP

The following commands are processed for BSC by the IPW$$BR BSC reader:

- SIGNON
- SIGNOFF
- START
- STOP
- GO
- SETUP

The remaining commands are given to IPW$$CM for processing. The main processor is responsible for de-coding each command and performing the processing necessary to cause appropriate action to the operator's request. Upon entry, a CP work area is acquired for a temporary command processor task. The CP work area is the primary work area for the task. The work area is initialized with the entry point addresses of the various command processor subroutines. To ensure that the permanent command processor task is not put in wait state due to virtual storage shortage, the first page of the pageable area is reserved at VSE/POWER initialization time as CP work area.

The command is then broken into its operands and each operand in turn is examined and edited; this is done by calling the FORMAT subroutine. A command table, containing the long and short form of the command verb, is used to determine the subprocessor to be entered. If the end of table is encountered or no subprocessor entry point address is present, the command is considered to be invalid.
Each entry in the table may have restriction indicators as follows:

- Command not allowed during VSE/POWER shutdown period
- Command not allowed at autostart time

If one of the conditions is true, the command is rejected. After processing the command, the caller's ECB, if present, is posted and the CP work area is released (temporary command processor only). A temporary command processor task is detached but the permanent command processor task is placed in inactive task. Control is then returned to task management service.

The following is a description of the various command processor subroutines.

MSG - subroutine: 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, flag SWFLAG3 must have been previously set to indicate that register 5 contains the address of that TCB from which this value is to be fetched.

The following functions are provided by the subroutine:

- Write message to system operator, and perform "connected message" processing using the WTO macro for messages to the local operator
- Write message to system operator and wait for his reply.
- Put message into remote (RJE) message queue.
- Handle AUTOSTART messages.
- Queue message destined for another node to appropriate node control block (NCB).

VERAUTH - subroutine: This subroutine checks if the issuer of the command is authorized or not. If the issuer has not enough authorization, message 1R85I or 1RA7I is issued.

The routine is called by means of the IPW$VCA macro instruction. The macro expansion has set up register 0 with a function code, describing the command to be executed. The function code is used as an index into appropriate tables which define for each command the authorization level required.

TCBSCAN - subroutine: 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.

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.

ATTACH - subroutine: 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 must have been previously set up by the calling routine in the dummy TCB located in the CP work area.

ASSIGN - subroutine: This subroutine updates the LUB/PUB tables of the VSE/POWER partition. The following actions are performed:

1. Locate the physical unit block (PUB) 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.

The subroutine calls the IPW$$LU routine by means of the IPW$ULP macro instruction.

UNASSIGN - subroutine: This subroutine resets the LUB assignment made for a task which cannot be attached due to a severe error. The subroutine calls the IPW$$LU routine by means of the IPW$ULP macro instruction.

INVDEV - subroutine: 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.

The following device type categories are supported:

- Tape devices
- Reader devices
- Punch devices
- Printer devices
- Diskette devices
- RJE control units
- PNET control units

QRINSPCT - subroutine: This subroutine determines whether the queue record of a queue entry meets the criteria set up by the calling routine. If so, the queue entry is marked eligible for modifying, deleting, releasing or holding.
Arguments may be:

- Class
- Jobname and jobnumber
- Job suffix number
- Generic jobname
- Binary RJE user id
- Local user id
- Password
- Current disposition, priority, forms id, system id, user id
- Originator destination and user id
- 'to' destination and user id
- Creation date
- Existence of time event scheduling information

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 entry meets criteria.
4 Queue entry does not meet criteria.

BINTODEC - subroutine: This subroutine converts a binary number into printable decimal format. Upon exit from this routine, field CONVDEC contains the decimal number in printable format, left-justified.

VQUEUEID - subroutine: This subroutine checks if the operand addressed by register 4 is a valid queue identifier. Valid queue identifiers are LST, PRT, RDR, PUN and XMT. Upon return field CLASSPTR contains the address of the class table associated to the specified queue or is zero when the operand is not a valid queue identifier. Field CLASSPCB contains the internal class type and field CLASSLC contains the maximum number of classes for the particular queue.

VTASKID - subroutine: This subroutine checks if the operand addressed by register 4 is a valid task identifier. Valid task identifiers are LST, PRT, RDR and PUN. Upon return, register 15 contains the return code:

0 Valid task identifier
4 Invalid task identifier

VPARTID - subroutine: This subroutine checks if the operand addressed by register 4 is a valid VSE/AF partition identifier. The GETFLD FIELD=PCEPTR returns for a valid partition identifier the PCE address in R1 and saves it into OTHPCE to be used by the calling command processor. If OTHPCE is zero the partition identifier is invalid. For a dynamic partition the GETFLD service is skipped because the PCE address is already in the command processor area of the TCB.

VERKEYOP - subroutine: 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:

- CCLASS
- CDISP
- CDUE
- CFNO
- CPRI
- CQNUM
- CRDATE
- CSYSID
- CUSER
- FNODE
- FULL
- FUSER
- LTAPE
- OUT
- REW
- TAPE
- TLBL
- TNODE
- TUSER
- XMTID
- CPAGES
- CCARDS
- LIMIT
- HOLD
- LOG

If the operand is none of above keywords, message 1R52I cccccccc OPERAND ## NOT SPECIFIED AS VALID KEYWORD is issued. If the keyword value is wrong, message 1R52I cccccccc INVALID SPECIFICATION FOR KEYWORD .... is issued. If the various keywords are inconsistent, message 1R52I cccccccc OPERANDS ARE INCONSISTENT is issued.
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. If the keyword value is valid, the appropriate argument list contained in the CP work area is updated accordingly.

CLASS - subroutine: 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. Each character of field "CPWCLAS" is checked until a blank character is found. If the maximum number of characters allowed is processed, message 1R87I cccccccc TOO MANY CLASSES, FIRST n PROCESSED is issued. If the character is invalid according to the task type, message 1R54I cccccccc CLASS x INVALID is issued. Upon return, register 15 contains the return code:

0 Valid class specification
4 Invalid class specification

FORMAT - subroutine: This subroutine edits and breaks out the operands of a VSE/POWER command. The positional operands are collected from the command control block associated with the command processor task and placed in the fixed format command area of the CP work area. This fixed format area is then used by the individual command processor routines. Each operand in turn is checked if alphameric and/or decimal. If so, appropriate flags are set. Decimal and hexadecimal operands are converted into binary format. Double quotation marks are replaced by single quotation marks. Keyword operands are split into keyword and keyword value. If the operand is embedded in quotes, the quotes are stripped off and an appropriate flag is set. If the operand is specified in hex notation, this is stripped off. Any +, - or * character preceding the operand is removed and the operand is flagged accordingly.

When an error is detected while processing the operands, one of the following messages is issued: 1R91I, 1R81I or 1R52I.

Upon return, register 15 contains the return code:

0 No error detected
4 Error detected during formatting

QRDISPCT - subroutine: This subroutine determines whether the queue record of a queue entry meets the criteria provided by the calling routine within the command processor work area or not. The command processor work area contains a queue record number identifying a queue record which is read by this routine. Once this queue record is read, the following fields of this queue record are compared with the values provided within the command processor work area:

- Queue id
- Jobname
- Jobnumber
- Job suffix number
- Password
- User id

The queue entry meets the criteria, if the following fields match to each other:

- Queue id
- Jobname
- Jobnumber
- Job suffix number (if not zero)

If the passwords do not match, the password within the work area must be a master password providing access authority. The user id specified within the work area must provide access authority: either it must match the to user id or/and the from user id or be a 'master user id' (hexadecimal zeros).

Upon exit from the subroutine register 15 contains following return code:

0 Queue entry meets the criteria.
4 Queue entry does not meet the criteria.

The entry point address of each subroutine is stored in the CP work area, thus it is available to all other command processor phases.

The following text describes the functions of the various command routines (phases) and its role in the overall command processor function:

PACCOUNT

This command is used to save the account file on tape, disk or cards or to empty the account file.

The operands are examined and a TCB for a 'save account' task is built. The TCB contains information about the medium where to save the account file. If the account file should be saved onto tape, the specified tape unit is assigned to the VSE/POWER partition. If a 'save account' task is already active, the command is rejected. Otherwise the save account task is attached.

PACT

The command is used to activate a drained transmitter or receiver task.

The operands are examined and saved into the CP work area. If no networking is supported or no connection exists to the specified node, the command is rejected. Otherwise the NCB is examined if the appropriate transmitter or receiver can be started. If the task is not in drained status, the command is rejected. Otherwise the drained flag is turned off in the appropriate NCB task entry and the PNET Driver is posted to create the transmitter/receiver task concerned.

PALTER Queue

This command is used to change one or more attributes of a job or job group.

The operands are examined and the QRINSPCT parameter list is built in the CP work area. The specified queue is then scanned for queue entries meeting the specified criteria. If such a queue entry is found, the queue entry is removed from the class chain, if necessary (only when class, priority, target destination, disposition, or forms id is to be altered) by means of the IPW$DQS macro instruction. If a non-dispatchable reader queue entry gets changed to a dispatchable one or the destination of a job in the transmission queue is changed in such a way that the entry gets queued into the local reader queue, indicate that change in the TCB for IPW$$TQ in order to calculate a new due date.
If a queue entry gets chained from the local queue (rdr, lst or pun) to the transmit queue, save the original disposition (meant for local processing) and set QRDP to the 'transmission disposition D'. Also save the original priority (meant for local processing) and set QRPY (transmission priority) to the same value.
If a queue entry gets chained from the transmit queue to the local queue (rdr, lst or pun), restore the original disposition meant for local processing, and restore the local processing priority.
The queue record is then updated and the queue entry is returned to the appropriate class chain by means of the IPW$AQS macro instruction when the queue entry was previously removed from the class chain. Otherwise the queue record is written back to disk by means of the IPW$AQS KEEP macro instruction. The following is a list of attributes which can be changed:

- Class
- Priority
- Disposition
- Target remote id
- Target destination
- Target user id
- Number of copies
- Compaction table name
- Due date information
- VM Distribution Code
- Forms Number
- Shared Processing SYSID
- User Information

PALTER LST|PUN,jobname,jobnumber,CQNUM=q-rec-number,SEGMENT=...

This command is used to segment job output by command.

The operands are examined and the QRINSPCT parameter list is built in the CP work area. If such a queue entry is found, the creating execution writer task is searched and if found, it is informed that the output must be segmented either on the next CARD or PAGE boundary or immediately. Information is passed by using TCVEB for posting and flags. The execution writer is then enabled for 1 DBLKGP from the DBLKGP cushion to allow segmentation in case of data file full (1Q38A).

PALTER Partition

This command is used to change the classes associated with a VSE/POWER controlled partition without stopping the partition concerned.

The operands are examined and the specified classes are checked for validity. The task selection list is then scanned to locate the execution reader task of the partition concerned. If found, the task class list table, contained in the TCB, is updated to reflect the new classes.

PBRDCST

This command is used to send a message to the specified recipient.

The operands are extracted and a nodal message record (NMR) is built in the CP work area. This NMR is then passed to the message distributor by means of the IPW$GMS TYPE=DIST macro instruction or in case of an ALLUSERS type message to the ALLUSERS message queue (via IPW$RMS). Depending on the return code given back an appropriate error/information message is issued.

PCANCEL

This command is used to terminate the printing of a VSE/POWER queue display or to cancel the execution of a VSE/POWER job currently running.

If 'STATUS' was specified, the TCB selection list is scanned for a matching 'print status' task. If found, the termination code 'S' is set in the TCB of the located task; otherwise the command is rejected.

If the first operand is a 'jobname', the TCB selection list is scanned for the task which processes the specified job. If found, the 'F' termination code is set in the TCB. The task is set dispatchable, if operator bound (waiting for operator reactivation). If the task is an execution reader task, all subordinary execution writer tasks are examined if operator bound. If so, the execution writer task is set dispatchable. All execution writers are enabled for 1 DBLKGP from the DBLKGP cushion to allow cancellation in case of data file full (1Q38A).

PDELETE Queue

This command is used to delete the specified job or job group.

The operands are examined and the QRINSPCT parameter list is built in the CP work area.

The specified queue is then scanned for queue entries meeting the specified criteria. If such a queue entry is found, the queue entry is deleted from the class chain by means of the IPW$DQS and IPW$FQS macro instructions.

PDELETE FCB

This command is used to delete VSE/POWER's FCB table.

The command processor root phase calls the PDELETE command processor where a validation check is performed. Then the command is checked for overspecification. If any error occurs an appropriate message is issued and control is returned to the command processor root phase. Otherwise the FCB table is locked by means of the IPW$RSR macro call and all 30 entries are reset to X'00'. Then the table is released by means of the IPW$RLR macro. Finally control is returned to the command processor root phase.

The address of the FCB table is located in CAT field CAFCTAB.

PDELETE MSG

This command is used to delete one or more messages from the ALLUSERS message queue.

The IPW$RMS macro instruction is called to delete the specified ALLUSERS message from the queue. If the command issuer is not authorized, the command is ignored.

PDISPLAY Queue

This command is used to display information on queued jobs.

The operands are extracted and the 'print status' parameter list is built in the CP work area. A 'print status' TCB is created and the parameter list is moved into it. If the queue display output should be printed on a printer, the specified printer device is assigned to the VSE/POWER partition. Finally the 'print status' task is attached.

The print status task will then scan the class chain(s) for the specified queue(s) and extract the status information required for a report to be printed on SYSLOG, passed to the remote terminal or node, or on the specified printer.

For a display of the CREATE queue or the DELETION queue the complete queue file is scanned to extract the status information.

The 'print status display' task displaying the BIGGEST queue entries requires a virtual storage 'collection area' to house the LIMIT number of queue records, in which they are gathered in descending order of used DBLKGP's from scanning the whole queue file (from queue record number 1 to MRQ#MAX). Finally each gathered record is displayed.

PDISPLAY PNET

This command is used to display information about the currently loaded network definition table (NDT).

The operands are extracted and the 'print status' parameter list is built in the CP work area. A 'print status' TCB is created and the parameter list is moved into the TCB. The print status task is then attached. The print status task will scan the network definition table and according the specified options, extract information from it to be either displayed on the system console, passed back to a remote node, or printed on the specified printer.

PDISPLAY VIO/QFL

This command is used to display the storage copy of the queue file residing in VIO or in partition GETVIS space.

PDISPLAY A

This command is used to display all active local, execution, RJE and networking tasks. One display line is shown for each such task.

The task selection list is scanned and each TCB in turn is examined if it is one of above mentioned tasks. If so, the task id, associated classes, information about the queue entry among others are displayed.

PDISPLAY EXIT

This command is used to display status information of VSE/POWER's currently loaded user exits.

The command attaches a display status task, which displays exit status information like name, type, size, load point and work area size.

The address of the exit table is located in the CAT field CAEXTAB. The exit table additionally contains the exits entry point address too, which is not reflected by the display.

PDISPLAY Q

This command is used to display the number of free queue records, the number of free DBLK groups and the account file full percentage.

PDISPLAY T

This command is used to display the time, date, pages fixed, SYSID, nodeid, and number of tasks.

PDISPLAY M

This command is used to display all tasks which are waiting for operator action.

The task selection list is scanned and each task in turn is examined whether the task is operator bound. If so, the task id and the reason why the message is waiting for operator action is displayed.

PDISPLAY MSG

This command is used to display all messages in the ALLUSERS message queue.

The IPW$RMS macro is used to obtain the first and subsequent messages from the ALLUSERS message queue. The message is then displayed as is.

PDISPLAY TRINFO

This command is used to display information of the

- telecommunication trace and
- task trace

PDISPLAY STATUS

This command is used to display the statistics status report about the current VSE/POWER session.

PDISPLAY DYNC

This command is used to display characteristics of classed contained in the currently active dynamic class table.

The operands are extracted and the 'print status' parameter list is built in the CP work area. A 'print status' TCB is created and the parameter list is moved into it. Finally the 'print status' task is attached. The print status task will then scan the dynamic class table, extract the information of the required classes and passes the information to SYSLOG, the requested user or builds a queue entry in the VSE/POWER LST queue.

PDISPLAY AUSTMT

This command is used to display the autostart statements, even if valid or invalid. If the FORMAT= statement was missing or invalid, the reply to message 1Q11D is also displayed (even if no autostart statement was read).

The autostart statements (and the reply to message 1Q11D) have been saved in IPW$$I2 and IPW$$I7: for each statement virtual storage has been reserved containing one statement and the eyecatcher 'AUTOSTMT:'. If not enough storage is available, the initialization is canceled.

PDISPLAY SPDEV(T)

This command is used to display the spooled devices for one or more partitions. If the operand SPDEVT is used, the device type used at IPL time and the device type in the PUB are displayed as well.

In order to display the device type used at IPL time, a translation table is used which is locally defined in IPW$$CD (DEVTYPTB).

PDISPLAY TASKS

This command is used to display information about all VSE/POWER tasks. Opposite to the PDISPLAY A command, the displayed information is 'internal' and thought to be helpful for VSE/POWER developper to solve VSE/POWER problems. Therefore feel free as VSE/POWER developper to adapt the displayed information whenever it is useful to improve debugging.

The additional operands for the operand A may be specified also for the operand TASKS. Therefore to process these additional operands the same code is used. The only difference is that per default all tasks are displayed for operand TASKS, even the so called 'internal tasks' like for example the PNET line driver, the cross partition master task and so on (see table TCBIDINT in IPW$$CD). To display only these 'internal' tasks the operand 'INT' may be used together with the TASKS operand.

To scan the TCB chain and to decide whether the TCB has to be displayed or not, the same code is used for the operands A and TASKS with a few special lines for the operand TASKS.

PDRAIN

This command is used to drain the transmitters or receivers specified and make them unavailable for further network activity until reactivated by a PACT command.

The operands are examined and saved into the CP work area. If no networking is supported or no connection exists to the specified node, the command is rejected. Otherwise the specified transmitter/receiver is addressed using the NCB task entry table and termination code 'E' or 'S' is set in the TCB. The NCB task table entry is flagged that the task is going to be drained.

PEND

This command is used to shutdown VSE/POWER.

The initiator TCB is changed to the terminator TCB, the termination type 'E' or 'S' is set in all existing TCBs and under certain conditions the task is set dispatchable. If a printer device is specified, SYSLST is assigned to the VSE/POWER partition.

In case of PEND or PEND cuu the following happens. The LCB chain is scanned and each signed-on remote operator is informed via message 1R99I that VSE/POWER is in shutdown. If RJE,SNA is active, the stop code is set in the SNCB and the SNA manager is posted to terminate. If networking is active, stop code 'E' is set in all NCBs and the PNET driver task, the TD-Subtask and the SD-Subtask are informed for shutdown. Next, the EDCB chain is scanned and a 'stop device' order record is built and queued at the tail of the order queue associated with each external device. The corresponding device service task is then posted to forward the order.

In case of PEND IMM or PEND IMM,cuu the following happens. The LCB chain is scanned and each signed-on remote workstation is stopped using the same stop codes as for the PSTOP FORCE command. If RJE,SNA is active, the same stop code as for the PSTOP SNA command without the EOJ operand is set in the SNCB and the SNA manager is posted to terminate. If networking is active, stop code 'S' is set in all NCBs and the PNET driver task, the TD-Subtask and the SD-Subtask are informed for shutdown. Next, the EDCB chain is scanned and the stop code is set: 'PSTOP with the FORCE operand' has been issued. The corresponding device service task is then posted to process the stop code.

In case of PEND FORCE, an IPW$CNC macro is issued without posting any termination type and thereby scheduling the VSE/POWER abnormal-end exit.

The following command hierarchy is used. Once PEND or PEND cuu has been entered, it may take its time till VSE/POWER comes to its end. Probably there is enough time to enter some more VSE/POWER commands, e.g. another PEND command. If another PEND is given, the message 1R88I OK is issued, although nothing else is done than the syntax checking (no new stop codes are set anywhere, no new posting of any task is done). Therefore the following happens:

1. If PEND has been issued and PEND IMM is issued as a 'second' PEND command, no special message is issued and the PEND IMM command is processed.
2. If PEND IMM has been issued and PEND is issued as a 'second' PEND command, no special message is issued and the PEND command is just syntax checked, no further processing occurs.
3. If PEND IMM has been issued and PEND IMM is issued as a 'second' PEND command, no special message is issued and the PEND IMM command is just syntax checked, no further processing occurs.

PFLUSH

This command is used to cancel the current activity on a device, locally or remotely attached, the execution of a job running in a VSE/POWER controlled partition, or the current transmission/receipt on a networking line.

The operands are examined if they specify a valid unit record device, partition, RJE task or transmitter/receiver task. If so, one of the following termination codes is set in the TCB:

'F' current queue entry to be deleted

'H' current queue entry to be bypassed, but not to be deleted.

The command is rejected, when the task is already at job boundary. In an execution reader is flushed, all subordinary execution writers are enabled for 1 DBLKGP from the DBLKGP cushion to allow flushing in case of data file full (1Q38A).

PGO

This command is used to reactivate a task which was waiting for operator response (task status 'O').

The task selection list is scanned for the task matching the specified operands. If found, the task status byte is examined. If the task in question is not operator bound, the command is rejected; otherwise the task is set dispatchable.

If the task is a device service task, a 'reactivate device' order record is built and queued at the tail of the order queue anchored to the EDCB which is associated with the task. The device service task is then posted to forward the order.

PHOLD

This command is used to take one or more VSE/POWER jobs out of the dispatchable state and put them in the hold/leave state.

The operands are examined and the QRINSPCT parameter list is built in the CP work area. The specified queue is then scanned for queue entries meeting the specified criteria. If such a queue entry is found, the queue entry is unchained from its class chain by means of the IPW$DQS macro instruction, the disposition in the queue record is changed from D to H and K to L; the queue record is then added back to the non-dispatchable class chain via the IPW$AQS macro instruction.

PINQUIRE

This command is used to display status information for RJE lines/sessions, external devices, and the network status:

The LCB (for RJE,BSC), SUCB/LUCB (for RJE,SNA), EDCB (for external devices) and/or the NCB (for networking) chains are scanned depending on the specified operands and status information is extracted.

- Not supported (no line table entry exits)
- Not initiated (no line control block or SNA control block exists)
- Inactive (no sign-on)
- Processing RJE-ID (sign-on)
- Status of each connection to other node
- Status of PNET TCP interface to IP partition at local node
- Status of PNET SSL interface to IP partition at local node

- Status of each transmitter/receiver
  - active
  - inactive
  - drained
  - halting (in process of closing down)
- Status of external device
  - active
  - waiting for work
  - waiting for operator reactivation
  - inactive
  - setup in progress

PLOAD NDT

This command is used to load the specified network definition table.

For a currently loaded network definition table the following checks are performed:

- Storage descriptor
- Local node name must match node name defined in master record
- 'own' node entry must be present in table
- Prime and alternate routing nodes must be defined
- VTAM application ids must be uniquely defined

If networking is not supported, the command is rejected. The command is also rejected when a severe error is detected while examining the NDT. When the NDT is correct, asynchronous service is used to perform the load operation. When at least one TCP node is defined, the 'PSTART TCPIP' command is triggered internally. When at least one SSL node is defined, the 'PSTART TCPSSL' command is triggered internally.

PLOAD EXIT

This command is used to load the specified user exit routine from the VSE/AF library.

The address of the loaded exit routine is saved in the CAT or PNCB, and the exit is set enabled. The possible specified length for a work area is anchored in the DMB or PNCB.

If networking is not supported but a NETEXIT/XMTEXIT should be loaded the command is rejected.

PLOAD DYNC

This command is used to load the library member DTR$DYNx.Z into the VSE/POWER partition for verification and optionally creation of the active dynamic class table with classes enabled for dynamic scheduling of partitions.

POFFLOAD

This command is used to save or re-load queue entries to tape or from tape.

The operands are examined and storage for the POFFLOAD task is acquired and initialized. The POFFLOAD parameter list is built in the TCB and the specified tape device is assigned to the VSE/POWER partition. The task is then attached by means of the IPW$ATT macro instruction.

PRELEASE

This command is used to take one or more VSE/POWER jobs out of the non-dispatchable state and put them in the dispatchable/keep state. If a reader queue entry gets released, indicate in the TCB for IPW$$TQ to calculate a new due date.

The operands are examined and the QRINSPCT parameter list is built in the CP work area. The specified queue is then scanned for queue entries meeting the specified criteria. If such a queue entry is found, the queue entry is unchained from its class chain and the disposition in the queue record is changed as follows:

- Disposition H (hold) to D (dispatch)
- Disposition L (leave) to K (keep)

The queue record is then added back to the dispatchable class chain. If a RDR queue entry is released which already once executed, a new VSE/POWER job number is assigned to that entry.

PRESET

This command is used to reset the in-execution flag for all jobs or output belonging to the SYSID that was specified in the command.

The command is also used to delete all NMR slots in the QCA belonging to the SYSID that was specified in the command.

PRESTART

This command is used to back or forward space a printer or punch device by the specified number of pages/cards.

The operands are examined and internal flags are set in the CP work area. The TCB chain is scanned to locate the task associated with the specified device. If found, the number of records to be skipped, and in the case of a 3800 Printer also the new copy group index to be used, is stored in the TCB of the local or remote writer task according to the operand specifications. The following index (type of skip) is set:

X'04'	Restart processing of the queue entry with specified record number.
X'08'	Skip as many records forward as specified.
X'0C'	Skip as many records back as specified.

If the task is a device service task, a 'restart device' order record is built and queued at the tail of the order queue anchored to the EDCB which is associated with the task. The device service task is then posted to forward the order.

PSEGMENT partition,cuu,...

This command is used to segment job output by command.

The addressed execution writer task is searched and if found, it is informed that the output must be segmented either on the next CARD or PAGE boundary or immediately. Information is passed by using TCVEB for posting and flags. The execution writer is then enabled for 1 DBLKGP from the DBLKGP cushion to allow segmentation in case of data file full (1Q38A).

PSETUP

This command is used to print one or more pages of list output with all printable characters replaced by 'X'.

The number of pages to be printed is stored in the TCB for the list task according to operand specifications. The task is posted dispatchable. The command is ignored when no list task exists or when the list task is not waiting for operator action.

If the task is a device service task, a 'setup device' order record is built and queued at the tail of the order queue anchored to the EDCB which is associated with the task. The device service task is then posted to forward the order.

PSTART

This command is used to create a TCB according to the operand specifications in the command (except for RJE,BSC lines or PNET). It attaches an execution reader task or a physical reader or writer task or RJE,SNA task or a PNET node to the network.

In case of a partition start it prompts the operator or the initiator task (if AUTOSTART) to supply the addresses of the devices to be spooled and builds the partition control block.

In case of a RJE,BSC start, the Line Control Block (LCB) is created and initiated. The activity byte is set to indicate line start for the Line Manager.

In case of a PNET, nodeid start, a node control block is created and formatted. If this is the first node being started, the PNET Driver task is created.
If this is the first SNA node being started, the VDCB is built and the PNET Driver is informed to open the interface to VTAM.

In case of the PSTART DEV command, a device service task is created and attached. The device service task is equipped with an external device control block (EDCB), which is chained as first element in the EDCB chain. A 'start device' order record is then built and queued at the tail of the order queue anchored to the EDCB which is associated with the task. The device service task is then posted to forward the order.

In case of PSTART TCPIP (accepted only by internal call), the PSTART task ensures by PNCB locking, that no previous same task is pending. Using the ATTACH communication TDCBSECB it verfies, that the TD-Subtask is not yet active and then it calls the ATTACH macro to give control to the TD-Subtask for entry into module IPW$$TD.

In case of PSTART TCPSSL (accepted only by internal call), the PSTART task ensures by PNCB locking, that no previous same task is pending. Using the ATTACH communication TDCBSECB in SDCB it verfies, that the SD-Subtask is not yet active and then it calls the ATTACH macro to give control to the SD-Subtask for entry into module IPW$$SD.

PSTOP

This command is used to stop either a partition controlled by VSE/POWER, a local reader or writer task, a networking or RJE,BSC line, or a SNA session.

In case of a partition stop, the TCB chain is scanned to locate the appropriate execution reader task. If found, stop code 'S' or 'E' is set in the TCB depending on the specified operands. If the task is not in L, M or C state, it is set dispatchable.

In case of a local reader/writer task, the appropriate termination code is set in the TCB of the task specified in the first operand of the command:

'S' stop immediately
'E' stop after processing of current queue entry

If the operands specify a PNET line, the termination code is set in the NCB and the PNET Driver is posted.

If the task is a device service task, a 'stop device' order record is built and queued at the tail of the order queue anchored to the EDCB which is associated with the task. The device service task is then posted to forward the order.

In case of PSTOP TCPIP (EOJ), the TD-Subtask is informed via TDCBACT1 flag to DETACH, when the last TCP node has stopped. In case of PSTOP TCPIP,FORCE, TD-Subtask cancellation (for entry into AB-Exit) is requested by the TREADY macro with cancel code X'08'. Thus IPW$$AT is informed about the reason of failure, namely 'due to PSTOP'.

In case of PSTOP TCPSSL (EOJ), the SD-Subtask is informed via TDCBACT1 flag (in SDCB) to request DETACH, when the last SSL node has stopped. In case of PSTOP TCPSSL,FORCE, SD-Subtask cancellation (for entry into AB-Exit) is requested by the TREADY macro with cancel code X'08'. Thus IPW$$AT is informed about the reason of failure, namely 'due to PSTOP'.

This command can also be used to 'unassign' a device (but only a printer, punch or tape device) which is assigned to VSE/POWER. This function has been made available, because it often happens at a customer's site that due to some software or hardware error a device is assigned to VSE/POWER but can not be unassigned by no means at all (except by issuing an IPL).

Two situations are processed differently:

1. If a VSE/POWER task does not exist which uses the device, the operand UNASSGN may be used to force an 'unassgn' of the device. Note that the specified cuu may be in two different places within a TCB, one due to the ccu1, the other due to the cuu2 according to the command
   PSTART LST,cuu1,X'cuu2'
   where the cuu1 is used for the printer and cuu2 for the tape from which the queue entry is read before it is printed. Therefore the scan for an existing task is not done by using the subroutine TCBSCAN in IPW$$CM, but by a routine within IPW$$CP. The 'unassign' is done by using subroutines in IPW$$LU.
2. If a VSE/POWER task does exist which uses the device (the scan is done as described above), the operand FORCE may be used to force an 'unassgn' of the device. This situation is processed like an 'immediate' stop condition. But only in 'special wait' situations this operand can be used, because it is assumed that for other 'normal' situations the 'normal' PSTOP command can be used. The 'special wait' situations are the following ones:

   a. waiting for virtual storage
   b. waiting for real storage
   c. waiting for operator reply
   d. waiting for tape, printer, punch I/O
   e. waiting for locked resource

   IPW$$CP posts the task dispatchable and sets indications within the TCB for the task dispatcher routine in IPW$$NU. The task dispatcher routine does not return to the caller, but gives control to IPW$$TR passing some indications within the TCB.

   IPW$$TR terminates as 'usual' the VSE/POWER subtask. 'Usual' means with the exception of issuing some messages and not releasing the storage area the address of which is passed within the TCB. The storage area not to be released is usually the area which contains the CCB (for unforeseen posting of the supervisor).

   The address of the storage area is updated by the routines which issue the I/O request: IPW$$PL, IPW$$PP, and tape I/O routine in IPW$$NU.

PVARY exit

This command is used to change the status (enable/disable) of a loaded exit routine or the status of the task trace.

If networking is not supported but a NETEXIT is concerned the command is rejected.

PVARY TASKTR
This command is used to enable/disable the VSE/POWER task trace for running.

PVARY DYNC
This command is used to disable or enable dynamic classes.

PVARY MSG
This command is used to influence the routing of VSE/POWER "I"nformational messages to 'hardcopy file only' (NOCONS) or to re-establish default routing (CONS), namely both to the console and hardcopy file.

PVARY MAXSAS
This command is used to increase/decrease the maximum (default=250) number of concurrent Spool Access Support (SAS) tasks allowed concurrently in a running system.

PXMIT
This command is used to route a command to another node in the network or to pass the command as is to the device driving system serving the external device specified in the command.

The operands are extracted and a nodal message record (NMR) is built in the CP work area. This NMR is then passed to nodal message service by means of the IPW$ICS REQ=ADD macro instruction. If the NMR forwarding failed (no connection established), appropriate error messages are issued.

If the command is destined for a device driving system, an 'xmit device' order record is built and queued at the tail of the order queue anchored to the EDCB which is associated with the task. The device service task is then posted to forward the order.

Command Processing Due to Operator Communication

Commands may be passed to VSE/POWER via the operator communication (OC) routine. This can be achieved by using the attention routine command MSG part,DATA=VSE/POWER command. The command specified after the operand DATA= is sent to the program running in the partition identified by part. In order to get such commands, the program must issue the VSE/AF macro STXIT together with the operand OC (operator communication).

VSE/POWER issues the STXIT macro in IPW$$I2. If the MSG command is issued before VSE/POWER has issued the STXIT macro, the message
1P60I NO ROUTINE LINKAGE is issued by VSE/AF.

VSE/POWER issues the STXIT macro with the following operands:

1. MSGPARM=YES to allow the passing of a VSE/POWER command after the DATA= operand of the MSG command
2. R1 containing the address of the OC-routine
3. R2 containing the address of the save area.

Both above addresses are pointing into the nucleus (IPW$$NU). This is done due to the following reasons.

This OC message facility should mainly be used whenever a problem occurs with VSE/POWER's processing. Therefore this support should

1. **not** alter any control blocks used by normal processing
2. **not** use those VSE/POWER macros which alter control blocks.

Thus the OC-routine does his own writing to the console (in IPW$$CM), and therefore uses its own buffer area (and CCB and CCW in error situations).

In order to avoid a lot of duplicate code, the commands which may be passed by the OC interface are the same as 'normal' VSE/POWER commands and these commands are processed by the same code (in IPW$$CM and IPW$$CD) as far as possible. In order to simplify the reuse of this code and have enough fields available for processing, the OC-routine has its own TCB which, however, is not chained into the normal task chain.

All these control blocks (TCB, CCB, CCW, and I/O buffer) must be pfixed and are defined in IPW$$NU, which is fixed, whereas both modules IPW$$CM and IPW$$CD are not pfixed.

The OC-routine in IPW$$NU just initializes some registers and branches to IPW$$CM. When returning from there, the VSE/AF macro EXIT with the operand OC is used to tell the VSE/AF supervisor that normal processing may continue.

Notes:

1. While the OC-routine is processing, the maintask of VSE/POWER is suspended. Even if for the OC-routine a page-fault occurs and the maintask is ready to run, the maintask does not receive control.
2. If for the OC-routine a page-fault occurs, the page-fault is passed to the page-fault appendage routine of VSE/POWER. In the page-fault appendage routine exists already code which omits the page-fault processing if register B points to an area not starting with 'TCB' (because of VTAM exit routines). Therefore the TCB of the OC-routine starts with 'STCB'.
3. Due to the fact that no control blocks should be altered and therefore most of the VSE/POWER macros should not be used, not all VSE/POWER commands can be issued via the message interface, but only those mentioned in table CPSTXOP defined in module IPW$$CM.

The reply messages to a PDISPLAY command are issued in a subroutine MSG000 in IPW$$CM. This subroutine MSG000 checks whether the TCB is used for the OC-routine and if yes, branches to routine CPSTXS00 in to order to isssue the reply messages.

Before processing a command, a dynamic workarea is retrieved (CPR000) and released (CPR960) in IPW$$CM, which is not done when the OC-routine is running: this workarea is allocated in module IPW$$CM in order to avoid the usage of a IPW$RSV and IPW$RLV macro. The setting of the pointers to this workarea and to other control blocks occurs in routine CPSTX000 in IPW$$CM, which also does the syntax checking of the command passed after the DATA= operand of the attention routine command MSG.

VSE/POWER Job Accounting

Account File Processing

Operations on the account file are performed by three functional routines:

- The build account record function (IPW$$BA), invoked by an IPW$PAR macro instruction,
- The put account function (IPW$$PA for C-K-D devices, or IPW$$PF for FBA devices) invoked by an IPW$PAR macro instruction,
- The get account function (IPW$$GA for C-K-D devices, or IPW$$GF for FBA devices) invoked by an IPWOAF, IPWGAR, or IPW$CAF macro instruction.

The build account record routine is called by the logical interface routines (IPW$$LO, IPW$$LR and IPW$$LW) to construct a LST, PUN, RDR or SPOOL account record from the information extracted from the queue record and associated work area. The account record is then passed to the put account record routine for writing to the account file.

The put account function routine will accept account records for the VSE/POWER partition and the partitions running under control of VSE/POWER (see Figure 58 on page 184). The account records (VARUNB format) will be written to the VSE/POWER account file under control of the account control block (ACB). The remaining file capacity is checked against a 20% limit. A warning message (1Q31I) is issued if the limit is exceeded. The message is issued in a fixed time interval of 60 seconds.

If the remaining capacity of the account file does not allow to store a presented record, message 1Q32A is issued and the operator is asked to save or to delete the account file and the task concerned is placed in a wait state (wait for ECB posting in account control block), until the account file is emptied by the save account task.

If a stop code 'S' (due to a PEND IMM command) has been set, the message 1Q81I is issued and the account processing is stopped immediately.

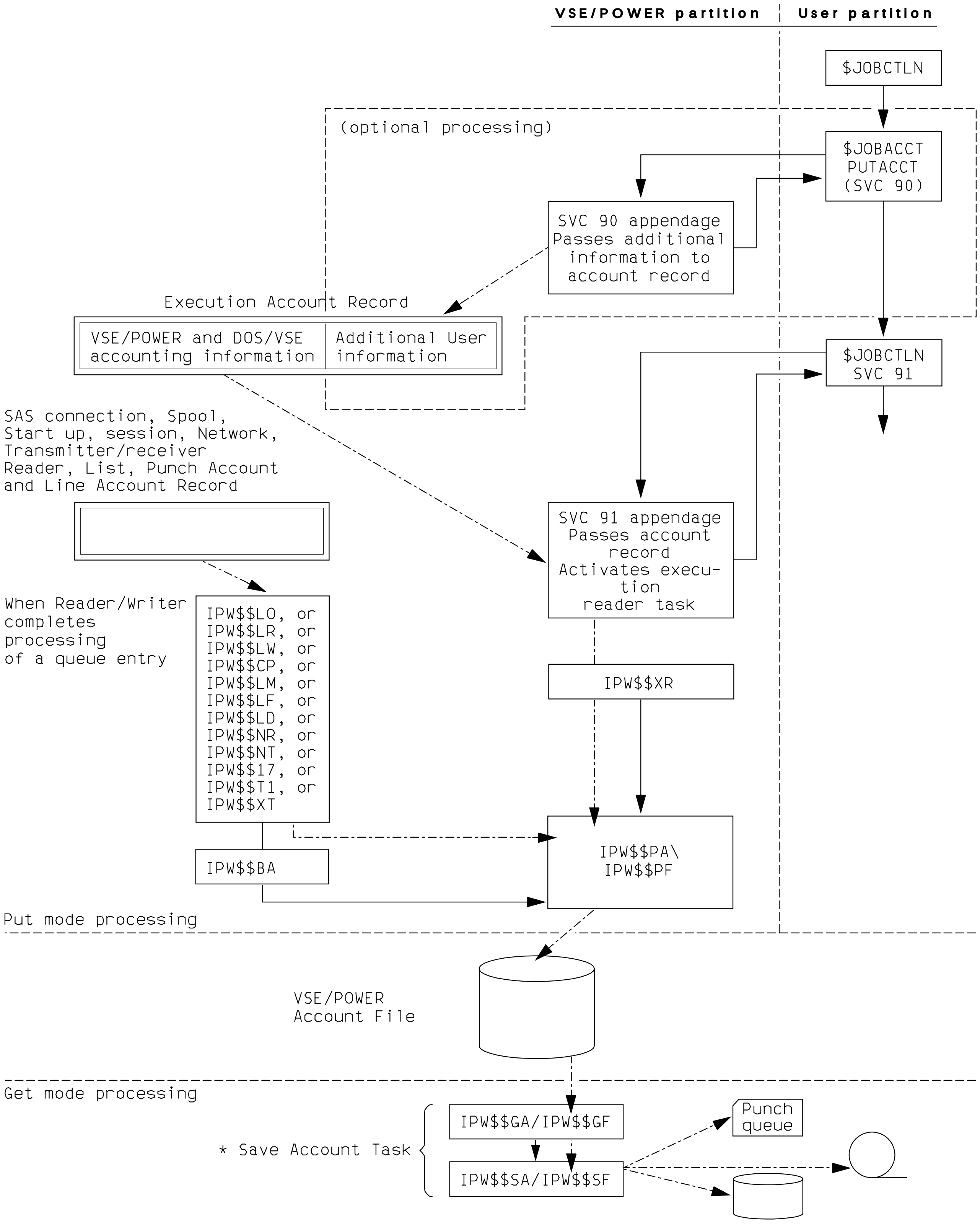


Figure 58. Relationship Between VSE/POWER and VSE/AF Job Accounting

The get account function routine, as used by the save account function, is broken down into three operations:

- Open account file for get mode processing, invoked by IPW$OAF macro instruction. This function is not supported for FBA devices.
- Get account record to retrieve the next sequential record from the account file, invoked by IPW$GAR macro instruction. This function is not supported for FBA devices.
- Close account file to restore the mode for put account record processing, invoked by the IPW$CAF macro instruction. For FBA devices, this function writes an SEOF (software end of file) to the account file.

Open Account File: The account control block is initialized for read operations (get mode) to retrieve the first record of the account file. During initialization, the account file is processed by the recovery routine, which seeks the last track written. There the last record is determined, and the next record is recorded as the location of the next available account record. If the system is running with a shared account file, then the next available account record is always updated and passed to the other shared systems via the master record.

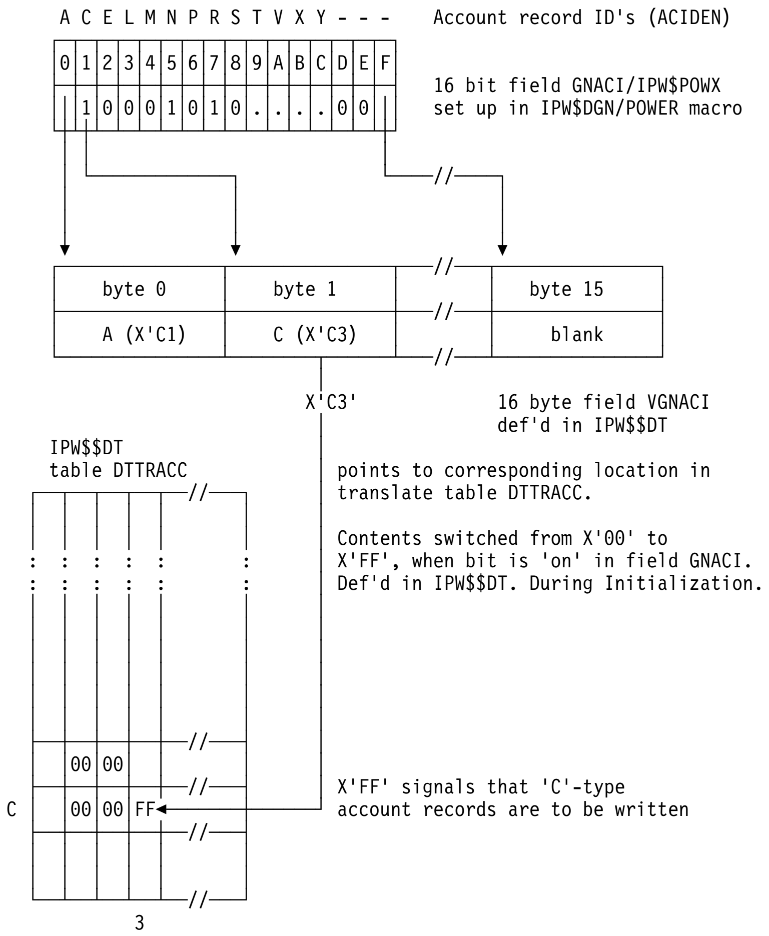
Close Account File: The account file records are erased by writing EOF records on each track for C-K-D devices or on the first block for FBA devices. The account control block is initialized to start on the first record in the account file. The task(s) waiting for posting of the ECB in the account control block are now allowed to continue processing.

Save Account Task: The save account task is attached by the command processor after a PACCOUNT command is given. The save account routine, IPW$$SA for C-K-D devices or IPW$$SF for FBA devices, is entered when the task gets control. Its purpose is to empty the account file, erase it, or save it on another storage medium (disk, tape, or punch queue).

User Specified Account Records: The system administrator can specify its own set of account record types by means of the POWER generation macro. This sets up a 16 bit field (field GNACI and field IPW$POWX), where each bit represents one specific type of account record. The account record types are arranged in ascending alphabetical order, that is the first bit at the left defines an account record of type 'A'(AFP), the next bit defines an account record of type 'C' (Spool-access-connect) and so on, and finally bit 12 defines the 'X'-type account record (Spool-access-operation account record. An exception is bit 13, which, if switched on, prevents creation of spool-access-operation account records which result from temporarily built $SPLnnnn queue entries. Bits 14, 15 and 16 are free for future account record types. The 16 bit table is defined the POWER macro and in VSE/POWER's generation table, whose layout is defined in macro IPW$DGN.

The 16 bit table has its equivalent in a 16 byte long character table (VGNACI, defined in IPW$$DT), in which the different id's of account records types are stored, that is, bit position 0 corresponds to byte 0 ('A) of this character table, bit position 1 corresponds to byte position 1 and so on. The relevant character is used to compute the offset of a byte in a translate table. This byte is then switched from X'00' to X'FF'. For example a '1' in bit position 0 selects byte position 0 in field VGNACI, which contains the character 'A'. The EBCDIC code for the character 'A' is hex 'C1' which points to offset 'C1' in the translate table. This byte is then switched from X'00' to X'FF' (Figure 59 on page 186).

Later when the decision is made whether or not a specific account record is to be written, the account record ID, obtained from field ACIDEN of the account record is checked against the corresponding location of the translate table. If X'FF' is found, the account record is written to IJAFILE.



Account record ID's (ACIDEN)

16 bit field GNACI/IPW$POWX
set up in IPW$DGN/POWER macro

16 byte field VGNACI
def'd in IPW$$DT

points to corresponding location in translate table DTTRACC.

Contents switched from X'00' to X'FF', when bit is 'on' in field GNACI. Def'd in IPW$$DT. During Initialization.

X'FF' signals that 'C'-type account records are to be written

Figure 59. User Specified Account Records, Data Relationship

VSE/POWER Networking Function

The VSE/POWER networking function, referred to in other parts of this manual as PNET, gives the user the ability to transmit jobs, output, messages or commands to other nodes in a network consisting of systems which support the network job interface (NJI).

A network is made up of one or more interconnected systems, called nodes. Each node in the network must have an image of the complete network obtained by means of the network definition table generated by the PNODE macro. The number of nodes that can communicate with each other is unlimited.

The transmission can be via binary-synchronous communication (BSC) lines or via a virtual channel-to-channel adapter (CTC) when running under VM or via a synchronous data line control (SDLC) line or via a TCP/IP link or via a TCP/IP SSL link. These types of connection are supported by one VSE/POWER PNET. Therefore, a network can consist of a node-A with BSC/CTC lines only communicating through a node-B with both BSC and SDLC lines to a node-C with only SDLC lines.

PNET TCP or PNET SSL support have been implemented such that they employ the PNET Line Driver function for native CTC communication passing CTC CCW-operations to the TCP/IP Driver Subtask (also called TD Subtask) or to the TCP/IP SSL Driver Subtask (also called SD Subtask), which provide the actual TCP/IP communication layer.

The VSE/POWER networking support is performed by following phases:

- IPW$$BS Buffer management
- IPW$$IN PNET Initialization
- IPW$$LD PNET Driver
- IPW$$LD1 PNET Driver BSC buffer processing
- IPW$$LD2 PNET Driver SNA buffer processing
- IPW$$LD3 PNET Driver Node activity control
- IPW$$LD4 PNET Driver VTAM activity control
- IPW$$LD5 PNET Driver subroutines
- IPW$$NC Composer
- IPW$$NK Compression/Decompression Routine
- IPW$$NM BSC I/O Manager
- IPW$$NP Presentation service
- IPW$$NR Receiver part 1
- IPW$$NR2 Receiver part 2
- IPW$$NT Transmitter
- IPW$$SE VTAM Exits
- IPW$$SR SNA Send/Receive Manager
- IPW$$S1 OPEN/CLOSE VTAM subtask
- IPW$$S2 Session establishment routine
- IPW$$S3 Session termination routine
- IPW$$TD TCP/IP Driver Subtask
- IPW$$TS TCP/IP Socket Support
- IPW$$SD TCP/IP SSL Driver Subtask
- IPW$$SS TCP/IP SSL Socket Support

PNET Initialization

If PNET= is specified in the POWER macro, the VSE/POWER initialization processor calls the PNET Initialization phase (IPW$$IN). This module

- Sets up the PNET master control block (PNCB) and stores its address in the CAT (CAPN).
- Sets up the TCP/IP master control Block (TDCB) and stores its address in the PNCB (PNCDTDCB).
- Sets up the TCP/IP SSL master control Block (SDCB) and stores its address in the PNCB (PNCDSDCB). Note, that throughout all VSE/POWER code, SDCB fields are addressed via the TDCB-DSECT!
- Loads all PNET phases in the pageable area after the last loaded phase for the local part.
- Loads after the PNET phases the user written XMTEXIT and/or NETEXIT if specified in the POWER macro and saves load address and possible work area length in PNCB.
- Loads the network definition table, in the GETVIS area, by invoking internally the PLOAD PNET command processor (IPW$$CLD), and waits for the completion of the following PLOAD actions:
  - load the NDT from the phase search chain
  - verify version '06.0' assembled under at least VSE/ESA 2.6
  - check and verify attributes of PNODE entries
  - pass TCP/IP related information from NDT to the TDCB
  - pass TCP/IP SSL related information from NDT to the SDCB
  - anchor new NDT in the PNCB (PNCBNDTN)
  - if at least one valid TCP node has been found in NDT, then invoke an internal command processor for 'PSTART TCPIP', which will (independent from PLOAD continuation) attach the TD Subtask as a VSE subtask to drive the interface from VSE/POWER to 'TCP/IP for VSE' from module IPW$$TD.
  - if at least one valid SSL node has been found in the NDT, then invoke an internal command processor for 'PSTART TCPSSL', which will (independent from PLOAD continuation) attach the SD Subtask as a VSE subtask to drive the interface from VSE/POWER to 'TCP/IP for VSE' for an SSL connection from module IPW$$SD.
- Acquires virtual storage for the temporary Node Attached Table (NAT) (in case running shared). This table is used as interface between the PNET driver and the timer task and contains entries for each node which was signed-on or signed-off since the last time interval. The timer task updates the NAT-table contained in the master record according to the entries found in the temporary NAT-table.

If an error occurs during PNET initialization, the appropriate error message is written to the system console operator and VSE/POWER is initialized without networking. If a problem with the exit routines occurred, PNET initialization continues, since the exit routines can be loaded dynamically later on with the PLOAD command.

PNET Driver

The PNET driver is the central control routine for the networking support in VSE/POWER. It schedules all SVC 0 communication requests for PNET BSC and CTC, passes PNET TCP requests to the TD Subtask or passes PNET SSL requests to the SD Subtask, handles communication error recovery and interfaces with VTAM for synchronous data link control (SDLC) communication. It controls all activity related to a node, like startup of the node, stopping of the node, etc., and controls the interface to VTAM. Nodes are identified by field NCBTYP whether they can be reached via a

- BSC link (no flag at all)
- CTC link (NCBCTCA flag)
- SNA link (NCBSNA flag)
- TCP link (NCBTCP and (!) NCBCTCA flag)

- SSL link (NCBSSL and (!) NCBCTCA flag)

Due to double flagging of TCP nodes the PNET Driver activities mentioned for PNET CTC also come true for TCP and SSL nodes (unless stated otherwise). The PNET driver is both event and time driven and contains several logic routines as described in the following.

PNET Driver Mainline (IPW$$LD): The PNET driver is attached by the first PSTART command entered for a remote node. It scans all existing nodes to determine what actions are to be performed, then executes the actions by invoking other routines. When all requested actions are completed it places itself into a VSE/POWER wait until posted by other VSE/POWER processors when there is more work to do.

First it checks if there have been buffers queued by either the VSE/POWER BSC/CTC channel-end appendage or by the TD Subtask emulating channel-end for PNET TCP or by the SD Subtask emulating channel-end for PNET SSL or by VTAM Send or Receive Exits, all on the buffer queue anchored to the PNET driver TCB. If so, the buffers are re-ordered from LIFO to FIFO sequence. Each buffer is then processed by IPW$$LD1, for BSC/CTC/TCP/SSL nodes, or IPW$$LD2, for SNA nodes.

When all buffers are processed, a check is made for the presence of VTAM related events and IPW$$LD4 is called to process them. Looping through the nodes (as represented by the chain of node control blocks), checks are made for actions requested for a node and if found the request is passed to module IPW$$LD3. A further check is then made for buffers to process before testing if the VTAM Interface may be closed in IPW$$LD4. The PNET driver waits to be posted for work to do and when dispatched continues the loop.

The PNET driver detaches itself when there are no active nodes any more and the VTAM interface is closed.

BSC/CTC/TCP/SSL Buffer Processing (IPW$$LD1): This module handles all buffers sent and received when the nodes are connected via a BSC line or CTC adapter or a TCP/IP link or TCP/IP SSL link. A multi-leaving (MLI) line discipline is used to control the line. Each buffer completed is checked for errors.

If the buffer is found to be in error or contains information that the remote node found an error, recovery is attempted. If the error is found to be unrecoverable or cannot be recovered within the retry limits, the node is flagged for termination.

If the buffer is found to be error free, it is analyzed to see if it contains data records or NJE stream control information. Data buffers are queued to the proper receiver.

If the buffer contained NJE stream control information then the proper actions, like flagging a receiver creation or flagging and posting of transmitters and receivers, etc., is performed.

The module then invokes the BSC I/O manager to execute the next Write-Read operation.

SNA Buffer Processing (IPW$$LD2): This module handles all buffers sent and received within an SNA session (established by module IPW$$S2 when the node was started). For SNA, a buffer also contains the RPL used to send or receive the data, which is then checked for an error and if found the node is flagged for termination and the buffer is freed.

If a SEND completes without error the buffer is freed after effecting any status change for transmitters or receivers as indicated in the buffer.

If a RECEIVE completes without error, the RPL is first analyzed for VTAM commands and if any is found then the node is flagged for termination. Non-command buffers are then analyzed in the same way as in IPW$$LD1.

Note: An SNA response may be sent when requested in the RPL.

Node Activity Control (IPW$$LD3): This module handles all events required for a node. An event is represented by an indication in the node's activity list. One or more events may be present at the same time and are processed in the following sequence:

- Timer Event for Node
- Notification of Transmitter/Receiver Detach
- Transmitter/Receiver Flow Control Event (SNA node)
- Delayed Buffer Processing (SNA node)
- Request to invoke VTAM Receive Manager (SNA node)
- Request to invoke VTAM Send Manager (SNA node)
- Request for Node Signon
- Request for Transmitter/Receiver Creation
- Request to Terminate Node
- Request to Signoff Remote Node
- Request to Terminate Session/Line
- Notification of Session Termination (SNA node)
- Request for Node Startup

The actions corresponding to these events are executed when the event is found to be present. The activity request is set up again for later execution when it cannot be currently executed because of resource shortage.

PNET - VTAM Interface Processing (IPW$$LD4): This module handles all events related to VTAM. More specifically it handles:

- Request to open VTAM interface (Open PNET ACB)
- Notification of VTAM OPEN Completion (End of Open PNET ACB)
- Notification of VTAM Termination by Operator or VTAM abend
- Request for Session by Remote Nodes
- Request to Quiesce Remote Node Session Request (SETLOGON QUIESCE)
- Request to end VTAM Interface (Close PNET ACB)

The module is called before activity request for individual nodes are processed and again after all activity for nodes are processed.

Note: The VTAM Interface is closed when no SNA nodes are active anymore and no VTAM related events pending anymore.

PNET Driver Common Subroutines (IPW$$LD5): This module is a collection of subroutines used by the modules above. It contains the following subroutines:

- Validation of Buffers Received (NJE Control Byte Validation)
- Error Logging on SYSREC
- Request Termination of Transmitters and Receivers
- Delete a Node from Node Attached Table (NAT)
- Attach a VSE/POWER Task for Session Creation/Termination
- Request Timer Events from Timer Services

PNET Node Operations

Starting PNET Nodes: This is requested by the PSTART PNET,node-id command, which - for SNA or TCP or SSL nodes - has to specify the remote node name only, and for BSC or CTC nodes the line address in addition. Hence the PSTART PNET command processor (IPW$$CPS) checks the currently loaded Network Definition Table (NDT) for the specified remote node name, creates a Node Control Block (NCB), and initiates node and link information in the NCB. Then the PNCB is locked to add the new Node Control Block to the NCB chain and the PNET driver task (if already attached) is alerted to start the initial link related processing for the chained NCB to achieve the 'signed on' state.

Starting a BSC Connection: If a BSC line is started, the PNET driver requests the Startup CCW Sequence (Disable, Setmode, Enable, Write SOH ENQ, Read) to be issued against the BSC line. The detailed flow of control when a BSC connection to another node is established, is shown in Figure 60 on page 193.

Starting a CTC Connection: If a CTCA line is started, the PNET driver issues a "stand-alone" SENSE-Commandcode CCW to determine the status of the adapter. At Sense I/O completion, the results are analyzed. If the sensed commandcode is zero, the CTC at the other end has not been started yet. VSE/POWER issues a CONTROL-READ CCW sequence and informs the system operator via message 1RC6I that the connection is pending. The CONTROL remains outstanding until the other side is started and issues its "stand-alone" SENSE. This process is designed to ensure that the two CTCAs will be synchronized with sense completing control, and read completing write. If the sensed commandcode indicates that a CONTROL is pending at the other side, VSE/POWER writes an SOH ENQ chained by a CONTROL-READ CCW. If, however, the sensed commandcode shows X'01' (write pending), a READ is issued to clear the adapter.

Starting a TCP Connection: The start of a TCP node is only accepted, if the TCP/IP interface (represented by the TD Subtask) is starting or already active. Only then the NDT is checked and the NCB is built with

- line = TCP
- NCBTYP = NCBTCP & NCBCTCA (!)
- a pointer to the related NDT entry
- the IP-address or pointer to the IP-name within the NDT
- an extra TCP/IP RECEIVE-ahead buffer of PNODE BUFSIZE

and the PNET driver task is posted. It handles the new NCB according to CTCA type and sets up a stand-alone SENSE-Commandcode CCW. According to TCP type, the called PNET I/O Manager module IPW$$NM does not request an SVC 0, but POST's the TCP/IP interface layer, represented by the TD Subtask. That attempts to

- CONNECT to the IP-host of the remote node,
- send the OPEN control record of the NJE TCP protocol,
- and obtain an ACK control record from the remote node.

Only then the TD Subtask alerts the PNET driver (of the CTC function layer) for completion of the stand-alone SENSE I/O with a sensed commandcode, as if CONTROL were pending at the other side. From now on processing continues as follows: initiated from the line driver (CTC layer), performed by the TD Subtask (TCP/IP interface layer), completion returned to the line driver, and so forth. For details on the TCP/IP interface processing refer to “TCP/IP Driver Subtask (TD Subtask)” on page 199.

Starting an SSL Connection: The start of an SSL node is only accepted, if the TCP/IP SSL interface (represented by the SD Subtask) is starting or already active. Only then the NDT is checked and the NCB is built with

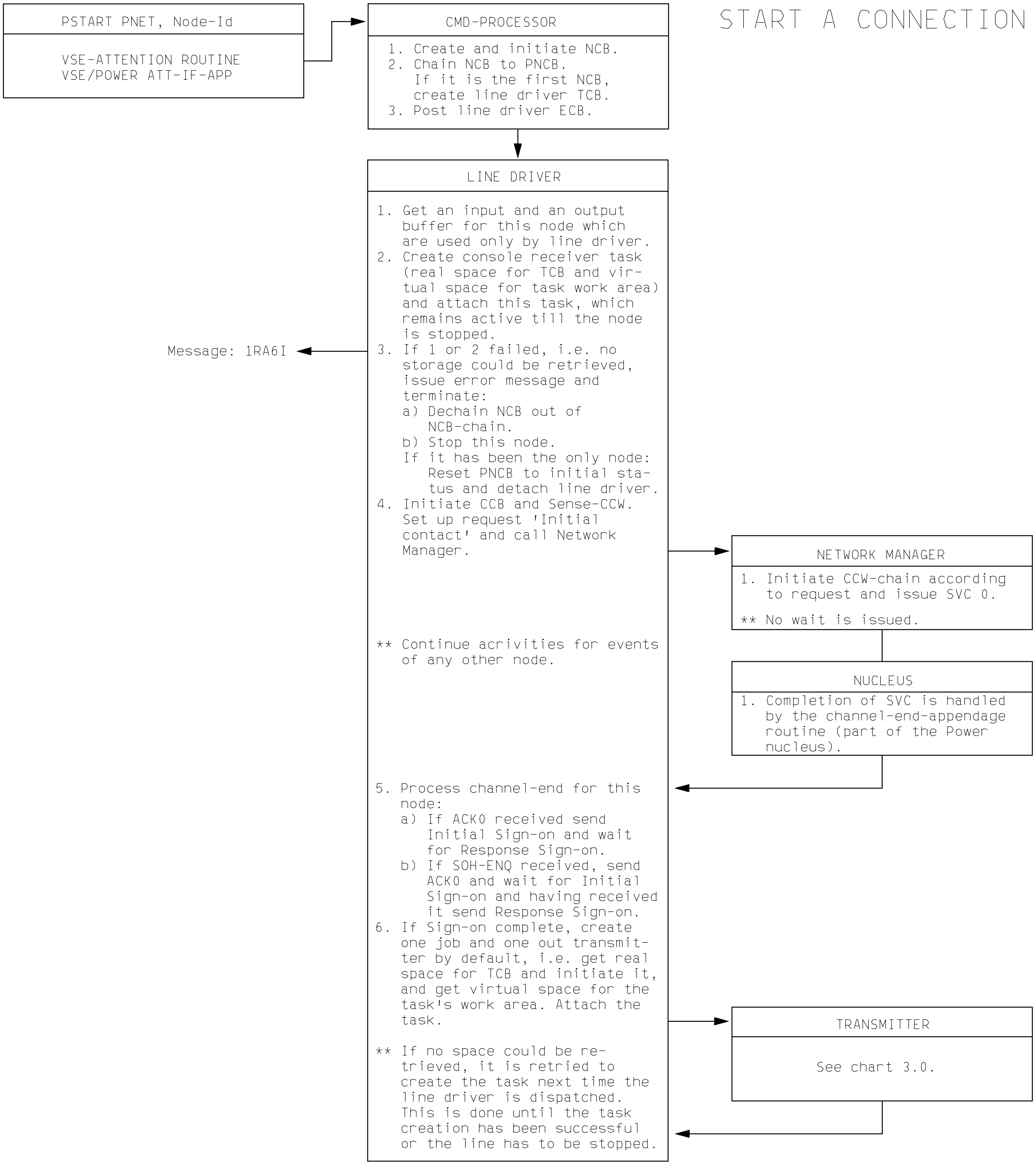
- line = SSL

- NCBTYP = NCBSSL & NCBCTCA (!)
- a pointer to the related NDT entry
- the IP-address or pointer to the IP-name within the NDT entry
- an extra TCP/IP SSL RECEIVE-ahead buffer of PNODE BUFSIZE

and the PNET driver task is posted. It handles the new NCB according to CTCA type and sets up a stand-alone SENSE-Commandcode CCW. According to SSL type, the called PNET I/O Manager module IPW$$NM does not request an SVC 0, but POST's the TCP/IP SSL interface layer, represented by the SD Subtask. That attempts to

- CONNECT to the IP-host of the remote node,
- send the OPEN control record of the NJE TCP protocol,
- and obtain an ACK control record from the remote node.

Only then the SD Subtask alerts the PNET driver (of the CTC function layer) for completion of the stand-alone SENSE I/O with a sensed commandcode, as if CONTROL were pending at the other side. From now on processing continues as follows: initiated from the line driver (CTC layer), performed by the SD Subtask (TCP/IP SSL interface layer), completion returned to the line driver, and so forth. For details on the TCP/IP interface processing refer to “TCP/SSL Driver Subtask (SD Subtask)” on page 223.



1. Create and initiate NCB.
2. Chain NCB to PNCB. If it is the first NCB, create line driver TCB.
3. Post line driver ECB.

1. Get an input and an output buffer for this node which are used only by line driver.
2. Create console receiver task (real space for TCB and virtual space for task work area) and attach this task, which remains active till the node is stopped.
3. If 1 or 2 failed, i.e. no storage could be retrieved, issue error message and terminate:
   a) Dechain NCB out of NCB-chain.
   b) Stop this node. If it has been the only node: Reset PNCB to initial status and detach line driver.
4. Initiate CCB and Sense-CCW. Set up request 'Initial contact' and call Network Manager.

** Continue acrivities for events of any other node.

5. Process channel-end for this node:
   a) If ACK0 received send Initial Sign-on and wait for Response Sign-on.
   b) If SOH-ENQ received, send ACK0 and wait for Initial Sign-on and having received it send Response Sign-on.
6. If Sign-on complete, create one job and one out transmitter by default, i.e. get real space for TCB and initiate it, and get virtual space for the task's work area. Attach the task.

** If no space could be retrieved, it is retried to create the task next time the line driver is dispatched. This is done until the task creation has been successful or the line has to be stopped.

1. Initiate CCW-chain according to request and issue SVC 0.

** No wait is issued.

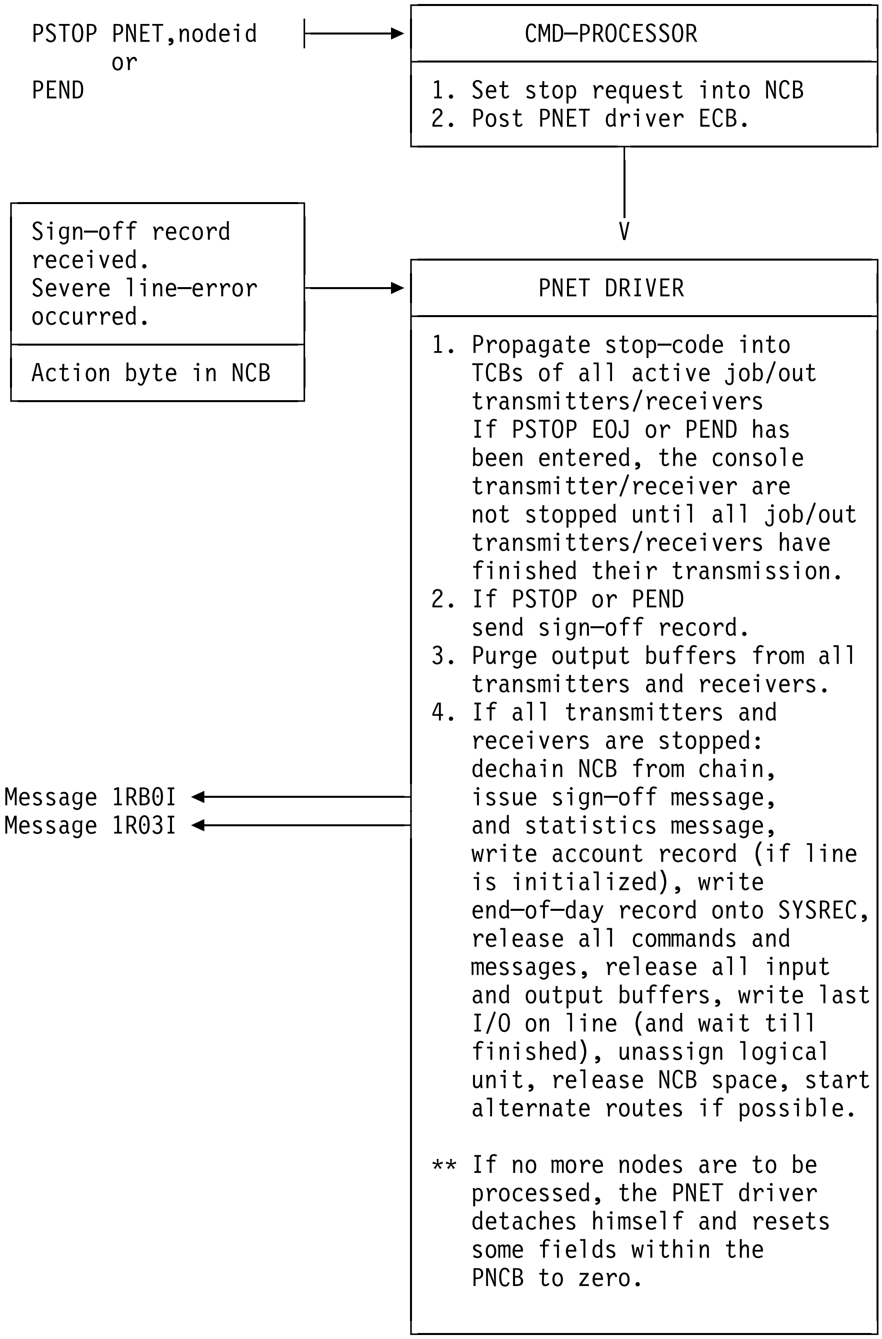
1. Completion of SVC is handled by the channel-end-appendage routine (part of the Power nucleus).

See chart 3.0.

Figure 60. Starting a PNET Connection - Shown for a BSC Link

Stopping PNET Nodes

Stopping a Node Connection: The control flow within VSE/POWER when a connection (of any link type) to another node is terminated, is shown in Figure 61 on page 194.



PSTOP PNET,nodeid
or
PEND

CMD–PROCESSOR

1. Set stop request into NCB
2. Post PNET driver ECB.

Sign–off record received.
Severe line–error occurred.

Action byte in NCB

PNET DRIVER

1. Propagate stop–code into TCBs of all active job/out transmitters/receivers
   If PSTOP EOJ or PEND has been entered, the console transmitter/receiver are not stopped until all job/out transmitters/receivers have finished their transmission.
2. If PSTOP or PEND send sign–off record.
3. Purge output buffers from all transmitters and receivers.
4. If all transmitters and receivers are stopped: dechain NCB from chain, issue sign–off message, and statistics message, write account record (if line is initialized), write end–of–day record onto SYSREC, release all commands and messages, release all input and output buffers, write last I/O on line (and wait till finished), unassign logical unit, release NCB space, start alternate routes if possible.

** If no more nodes are to be processed, the PNET driver detaches himself and resets some fields within the PNCB to zero.

Message 1RB0I
Message 1R03I

Figure 61. Control Flow when Stopping a PNET Connection

PNET BSC/CTC/TCP/SSL I/O Manager

The I/O manager is responsible for issuing all I/O's for a PNET BSC or CTCA line and for passing PNET TCP or PNET SSL communication requests from the CTC layer to the TD Subtask (operating in IPW$$TD) or to the SD Subtask (operating in IPW$$SD). Whenever an I/O is to be initiated, the PNET driver requests this by indicating the type of request in the node control block (NCB) addressed by register 1.

Depending on the request code a channel program is constructed in the node control block and the I/O is started via either a direct SVC 0 (BSC/CTC) or via alerting the TD or SD Subtask by POST'ing the TDCBECB in the TDCB or SDCB. In addition the block control byte (BCB) sequence count is updated accordingly and the just sent buffer is completed with the current FCS bytes. No wait is performed after the SVC 0 and immediate return is made to the PNET driver.

The request code consists of a main request (first four bits) and a sub request (last four bits). Figure 62 shows the various CCW sequences that can be built by the I/O manager.

X'01' startup sequence

- Disable
- Setmode
- Enable
- Write SOH ENQ
- Read response into PNET driver input buffer

For CTC, the CCW sequence consists of:

- Sense
- Write SOH ENQ
- Control
- Read response into PNET driver input buffer

X'02' retry startup sequence

- Sense
- NOP
- Enable
- Write SOH ENQ
- Read response into PNET driver input buffer

For CTC, the CCW sequence consists of:

- Sense
- Write SOH ENQ
- Control
- Read response into PNET driver input buffer

X'03' CTC read only sequence

- Read

X'04' disable sequence

- Disable

X'05' retry sequence

- The last issued channel program is re-issued, if the last request was a NAK, the last non NAK is issued instead.

X'06' CTC stand-alone read sequence

- Control
- Read response

X'07' read only sequence

- The last issued channel program is scanned for a READ CCW. If found the READ CCW is moved up to the first position of the channel program and is re-issued (without chaining).

Figure 62 (Part 1 of 2). PNET BSC and CTC CCW Sequences

X'08' expedited flow sequence

- Write text from PNET driver output buffer
- Write DLE ETB
- Read response / text
  If a free input buffer is available or a new buffer can be obtained from the VSE/POWER storage pool this buffer is used as input buffer and the READ CCW is updated accordingly. Otherwise, the PNET driver input buffer is used and the wait-a-bit flag is set.

X'09' CTC sense only sequence

- Sense

X'10' write - read sequence

- Write text / empty buffer
- Write DLE ETB
- Read response / text
  The current status of the to-be-sent output queue is examined and depending on its condition either an empty buffer is sent, the next output buffer transmitted or an 1,5 second delay initiated. If a wait-a-bit was just received an empty buffer is sent to acknowledge the received input buffer. If a free input buffer is available or a new buffer can be obtained from the VSE/POWER storage pool this buffer is used as input buffer and the READ CCW is updated accordingly. Otherwise, the PNET driver input buffer is used and the wait-a-bit flag is set.

  A stand-alone write request is forced when the sign-off record is to be sent for a CTC.

X'15' CTC stand-alone write sequence

- Sense
- Write text

X'20' NAK sequence

- Write NAK
- Read response / text
  If a free input buffer is available or a new buffer can be obtained from the VSE/POWER storage pool this buffer is used as input buffer and the READ CCW is updated accordingly. Otherwise the PNET driver input buffer is used.

X'21' ACK0 sequence

- Write ACK0
- Read response / text
  If a free input buffer is available or a new buffer can be obtained from the VSE/POWER storage pool, this buffer is used as input buffer and the READ CCW is updated. Otherwise the PNET driver input buffer is used.

Figure 62 (Part 2 of 2). PNET BSC and CTC CCW Sequences

PNET TCP Interface to TCP/IP

Establishing the Interface to TCP/IP: At PNET initialization by IPW$$IN, when the internal PLOAD PNET command is launched, or at any later time, when an operator requests (re)-loading of the Network Definition Table by the PLOAD command, module IPW$$CLD checks the new NDT for at least one valid TCP node and invokes then the PSTART TCPIP command. The IPW$$CS command processor accepts this startup attempt for the TD Subtask (representing the TCP/IP interface) only by a VSE/POWER internal request and does the following:

1. Lock/unlock the PNCB to record the temporary PSTART command task in TDCBATCB as a serial resource, which intends to attach the TD Subtask. When another PSTART TCPIP task is pending, new attempts are rejected.
2. Check the VSE ATTACH/DETACH communication TDCBSECB of the TD Subtask for subtask still down or already alive. If the subtask has been started previously and no stop code is pending, then terminate the PSTART task. If subtask is active, but stop codes are pending in TDCBSTA1 or TDCBACT1, then the PSTART task checks periodically until the TD Subtask has terminated in order to attempt a new attach request.
3. Use VSE ATTACH to give control to the TD Subtask for module IPW$$TD.
4. Finally lock/unlock the PNCB to clear the PSTART task pointer in TDCBATCB and terminate this command processor.

The TD Subtask is attached with the following processing attributes, which outline the subtask's embedding within the surrounding VSE/POWER functions and services. The TD Subtask

- calculates its entry point into module IPW$$TD after
  1. the VSE/POWER storage descriptor,
  2. the VSE subtask save area,
  3. the Subtask-id 'IPW$$TD', and
  4. the Subtask abnormal termination save area
- uses VSE/POWER maintask's ABEXIT in IPW$$AT
- drops protection key 0 (established during attach) in order to run in parallel mode
- provides local save area for requesting Idumps in flight using VSE/POWER's IPW$IDM support for a VSE Subtask
- provides local Message Control Block for requesting IPW$$MX message modification for IPW$$MM defined messages
- uses macro IPW$GTO to request message support from TCP/IP Service Module IPW$$TS
- uses macro IPW$TTM to request STXIT timer interval support from TCP/IP Service Module IPW$$TS
- uses macro IPW$ITP to request EZASMI Socket calls from TCP/IP Service Module IPW$$TS
- uses VSE macro WAIT TDCBECB to be activated by the PNET Driver from the VSE/POWER CTC layer
- uses VSE macro POST PAEB (with Line Driver task ecb posted additionally) to activate the VSE/POWER CTC layer from the TCP/IP interface layer
- uses Test-and-Set instruction to lock the PNCB for sharing the NCB chain with the Line Driver task, the SD Subtask, and the PSTART PNET command processor task - marking PNCB lockword with 'TCP', when owned by TD Subtask
- uses Compare-and-Swap instruction to add an 'I/O completed' CTC input buffer to the Line Driver Channel End Queue for sharing this resource with the Line Driver task, the SD Subtask, and the I/O Supervisor Task
- cannot use macro IPW$GTE to reserve an entry of the telecommunication trace area or to call trace area dumping, instead ...
- uses Test-and-Set instruction to lock the Trace Information Block (TIB) for sharing line trace entries with the PNET Line Driver task, the SD Subtask, and the RJE Line Manager task - marking TIB lockword with 'TCP', when owned by TD Subtask

- uses Test-and-Set instruction to lock the Asynchronous Service Anchor Block (ASAB) when sharing trace area dumping with the PNET Line Driver task, the SD Subtask, and the RJE Line Manager task - marking ASAB lockword with 'TCP', when owned by TD Subtask
- calls macro IPW$IAS to invoke the Dump Subtask of IPW$$AS for trace area dumping. Selected parts of this module have been made sensitive on 'being called by VSE/POWER task or by TD Subtask'

Controlling the TCP/IP Interface: Once the TD Subtask has been attached (see message 1RTMI) it remains active, even when another NDT is loaded lateron without any remote TCP node. The 'active' state can be interrogated by either command

- STATUS F1 (assuming VSE/POWER in F1), presenting the 'IPW$$TD' subtask of F1
- PINQUIRE NODE=local-node, presenting '1R56I TCP/IP: ...' information.

The TD Subtask is only terminated by external request

- at normal VSE/POWER session termination time through the PEND command, which sets TDCBACT1 flags. Then, after all TCP nodes have been stopped, the TD Subtask requests DETACH. When however all (non core) VSE/POWER tasks have terminated and the termination processor IPW$$T1 still finds the TD Subtask communication TDCBSECB with 'alive' indication, there is safety code, so that the termination task requests DETACH for the TD Subtask.
- during a VSE/POWER session through
  - the PSTOP TCPIP (EOJ) command, which informs the TD Subtask for termination processing as if PEND were given.
  - the PSTOP TCPIP,FORCE command. This format is only intended for halting the TD Subtask abruptly, in case it seems to 'hang'. Hence the IPW$$CP stop command processor requests subtask cancellation by the TREADY call (with cancel code X'08' = 'due to PSTOP), which leads to AB-Exit processing in IPW$$AT.
  - an abnormal Subtask termination (e.g. program check) leading to AB-Exit processing in IPW$$AT, which has as well standard VSE subtask tidy-up steps (message 1Q2CI and Idump) as a subtask type specific step. For the TD Subtask the TCP/IP related tidy-up routine is called, which is located in module IPW$$TD, and which terminates all TCP node processing and the interface to TCP/IP for VSE. Upon return to IPW$$AT the TD Subtask finally requests DETACH.

TCP/IP Driver Subtask (TD Subtask)

Overview: The TCP/IP Driver subtask or TD Subtask processes as a VSE/ESA subtask using the modules IPW$$TD and IPW$$TS. IPW$$TD is the main routine and calls IPW$$TS for following purposes:

1. Issue a socket call
2. Test the returncode of a socket call
3. Issue a message
4. Use timer services:
   a. Initialize timer services
   b. Set up a timer interval
   c. Process expired timer intervals
   d. Cancel a timer interval
   e. Wait a bit

In addition following modules, which are used mainly by the VSE/POWER maintask, are used by the TD Subtask as well:

IPW$$AT Process abnormal termination

IPW$$AS Write storage trace entries to dump libraries

The TD Subtask starts its processing after it has been attached by the VSE/POWER maintask when a network definition table is loaded with at least one TCP node.

The TD Subtask ends its processing

1. In normal situations due to any PSTOP or PEND command by detaching itself using the VSE/ESA macro DETACH
2. In abnormal situations by returning to IPW$$AT after some cleanup processing

The main purpose of the TD Subtask is to process requests concerning a TCP node:

1. Translate an I/O request consisting of several CCW's (built according to the CTC protocol) into socket calls and call the TCP/IP layer by using the macro EZASMI
2. Process the completion of a socket call
3. Pass the received data of a socket call into the buffer of a read CCW and queue this buffer to the channel-end-queue which is processed by the PNET Driver
4. Process any normal or abnormal stop condition

Interfaces and Operation Layers: The TD Subtask translates an I/O request (which has been built by the PNET Driver) into socket calls. To issue the socket call, the TD Subtask communicates with the product TCP/IP for VSE/ESA parts of which run in its own partition (usually F7). Figure 63 shows the three layers involved in this communication.

```
                          NODE A                                           NODE B

     I                   II            III     (IV)    III             II                I

  PNET TCP                           TCP/IP          TCP/IP                            PNET/TCP
 Line Driver  <---> TD Subtask <--->  part.--INTERNET--part.  <---> TD-Subtask <--->  Line Driver

|---P O W E R ---PARTITION----|    |TCP-PART|        |TCP-PART|    |---P O W E R----PARTITION-----|
```

Figure 63. TD Subtask - Three Operation Layers for PNET TCP support.

The TD Subtask, however, does not communicate directly with TCP/IP for VSE. Instead, it communicates with the LE/VSE C socket interface, which directly communicates with TCP/IP for VSE. To address the LE/VSE C socket interface, VSE/POWER uses an application interface (EZASMI macro) which has been introduced with VSE/ESA 2.5. Some routines of these three components are loaded into the VSE/POWER partition, some into the SVA.

The communication between the VSE/POWER partition and the TCP/IP partition (F7 in the system setup distributed by VSE/ESA) is done by routines of the product TCP/IP for VSE/ESA which uses the VSE/ESA XPCC interface.

Posting Events: The TD Subtask gets posted by the

1. Command processor (IPW$$CPS) to start its processing (when being attached)
2. VSE/ESA Supervisor due to an expired timer interval set up by the TD Subtask itself
3. PNET Driver (IPW$$NM) to translate an I/O request
4. PNET buffer services (IPW$$BS) to complete an outstanding I/O request because an output buffer has been put into an empty buffer queue. This can be caused by one of the following tasks:
   a. by a transmitter task (even by a console transmitter)
   b. by a receiver task (queuing a PGR, NPGR, or EOT record)
   c. by the PNET Driver task (queuing a network control record, for example a SIGNON or SIGNOFF record)
5. Command processor (IPW$$CP) to terminate its own processing (due to a PSTOP TCPIP command).

Note however:

1. If a PSTOP PNET,nodeid command without the FORCE operand has been entered, the connection to a node is terminated properly by sending a SIGNOFF record to the remote node, which means the TD Subtask gets posted by the PNET Driver (IPW$$NM) due to a new I/O request.
2. If the command PSTOP PNET,nodeid,FORCE (with the FORCE operand) has been entered, the PNET Driver is posted but not the TD Subtask. The connection is terminated unproperly, no SIGNOFF record is sent to the remote node. The TD Subtask processes the immediate stop request, when posted due to its own timer services.
3. If the command PSTOP TCPIP,FORCE (with the FORCE operand) has been entered, the TD Subtask is cancelled for entering the abnormal termination routine IPW$$AT.
4. If the PSTOP TCPIP command without the FORCE operand has been entered, the TD Subtask waits till all nodes are stopped:
   a. If a node is signed-on, the node must be stopped explicitly by issuing a PSTOP PNET,nodeid command.
   b. If a node is not signed-on, the connection is terminated unproperly, no SIGNOFF record is sent to the remote node.

Addressing Mode: The TD Subtask runs usually in 24-bit addressing mode. Running within module IPW$$TD, the TD Subtask runs for a few instruction in 31-bit addressing mode when addressing data received by following socket calls:

GETHOSTBYADDR receiving a logical hostname according to the received binary IP-address via a socket call CONNECT.

GETHOSTBYNAME receiving a binary IP-address according to the logical hostname specified in the network definition table (NDT)

Running within module IPW$$TS, the TD Subtask runs in 31-bit addressing mode when issuing a socket call.

Processing as Server and Client: Within a TCP/IP network, an application may run as a client or server. The client is the application which issues a CONNECT request, the server is the application which issues the socket calls BIND, LISTEN and ACCEPT and processes a CONNECT request of another client. The client is said to run in 'active mode', whereas the server is said to run in 'passive mode'.
The TD Subtask processes as a server and as a client. Whenever a PSTART command for a node has been entered, the TD Subtask issues a CONNECT request and acts as a client. If the remote node does not answer or rejects the CONNECT request, the TD Subtask suspends its active mode for a while (usually 2 minutes). During this time a CONNECT request can be received from the remote node, and the TD Subtask acts then as a server. In order to process incoming CONNECT requests from remote nodes, the TD Subtask issues an ACCEPT request at the beginning of its processing. Whenever the ACCEPT request signals an incoming CONNECT request, the TD Subtask processes this request and thereafter issues a new ACCEPT request for more CONNECT requests. As soon as a connection to a remote node has been established successfully, all connections are internally flagged as processing in 'active mode'.

Processing the 'Initial Contact': The usage of a TCP/IP network as a physical layer for a logical NJE network has been first implemented by RSCS. Hence VSE/POWER implemented the same rules:

1. A connection is established according to the CTC protocol, which means at the beginning of the connection the BSC characters SOH-ENQ and DLE-ACK0 are exchanged.

2. All data exchanged according to the CTC protocol are blocked into a TCP/IP block using the following structure:

   TTB 8 bytes describing a block of NJE data

   TTR 4 bytes describing a record of NJE data

-- n bytes containing NJE data

TTR-EOB 4 bytes describing the end of a block of NJE data

At this point, however, one TCP/IP block contains only one record of NJE data

3. Before starting to exchange data according to the CTC protocol, an 'initial contact' is established, namely two control records, an OPEN and an ACK or NAK control record, are exchanged to verify that the two nodes adhere to the NJE protocol. The OPEN control record is exchanged first, whereas the ACK or NAK control record is sent as response to the OPEN control record. The ACK is sent as a positive acknowledgement to continue with the connection, whereas the NAK is sent as a negative acknowledgement to stop the connection. All control records contain the following 33 bytes:

   a. 8 bytes describing the type of the control record
   b. 8 bytes describing the FROM NJE nodename
   c. 4 bytes describing the FROM IP-address
   d. 8 bytes describing the TO NJE nodename
   e. 4 bytes describing the TO IP-address
   f. 1 byte describing a return-code, which is used only for a NAK control record

Details about the TCP/IP frames and control records are described in the macro IPW$DTP, starting at the lables TCPTTB, TCPTTR and TCPCTRL.

Processing I/O Requests: The I/O requests processed by the TD Subtask are built according to the CTC protocol by the PNET Driver (IPW$$NM). Instead of issuing a START I/O request, the PNET Driver updates the status 'I/O request to be processed' (NCBTPS3S) and posts the TD Subtask. When the TD Subtask gets control, it loops through the NCB-chain and finds the I/O request to be processed (NCBTPS3S). To flag the I/O request 'complete', the TD Subtask updates the CCB and queues a buffer to the channel-end-queue anchored in the PNET Driver TCB (TCBQ) using Compare-and-swap (like the Channel-End-Appendage routine for BSC- and CTC-nodes, and the SD Subtask). The TD Subtask updates the CCB always with channel and device end, which means the PNET Driver will never issue any special I/O request for recovery purposes. If any error occurs, the TD Subtask informs the PNET Driver (IPW$$LD1) by setting NCBTPS1E, which causes the node to be stopped on the NJE layer.

The PNET Driver issues only the following CTC I/O requests for a TCP node:

1. Stand-alone SENSE CCW

   The I/O request consists of one SENSE CCW only. This request is issued only as the first request when starting the connection for a node. It is used to synchronize the I/O requests with the remote node. The input for a SENSE CCW is one byte, the command code pending on the remote node. The TD Subtask returns only two different command codes:

   a. X'07' (CTC Control), if the remote node did not yet issue an I/O request. The PNET Driver issues as response to this sense byte an I/O request containing four CCWs, a SENSE, WRITE, CONTROL and READ CCW. The WRITE CCW sends an SOH-ENQ to the remote node and the READ CCW should receive an DLE-ACK0 from the remote node.
   b. X'00', if the remote node issued already an I/O request. The PNET Driver issues as response an I/O request containing two CCWs, a CONTROL and READ CCW. The READ CCW should receive an DLE-ACK0 from the remote node.

   Although for a CTC line other values than these two may be returned for a SENSE CCW, the TD Subtask restricts itself to these values which are sufficient to handle the two different events, whether the remote node issued already an I/O request or not.

2. A READ only request

The I/O request consists of two CCWs, a CONTROL and READ CCW. This I/O request is issued only as a response to X'00' received by a stand-alone SENSE CCW. The READ CCW should receive an SOH-ENQ from the remote node.

3. A WRITE only request

   The I/O request consists of two CCWs, a SENSE and WRITE CCW. This I/O request is issued only when the connection to the remote node has to be stopped according to the NJE protocol: a SIGNOFF record is sent to the remote node without waiting for any response.

4. A WRITE/READ request

   The I/O request consists of four CCWs, a SENSE, WRITE, CONTROL and READ CCW. This I/O request is issued in all cases except the three cases described above.

The TD Subtask performs following actions for the above described I/O requests:

1. Stand-alone SENSE CCW

   This I/O request is not completed before the initial contact (see “Processing as Server and Client” on page 201) has been done successfully, which means the OPEN and ACK control records have been exchanged. Once a PSTART command has been entered, the TD Subtask issues a CONNECT request to start a TCP/IP connection in active mode. If the CONNECT request fails or a NAK control record is received instead of an ACK control record, the TD Subtask retries the the CONNECT request, usually every 2 minutes, or completes the initial contact via the passive mode in case the remote node started the connection with a CONNECT request.
   This means that the PNET Driver (IPW$$LD3) has been changed:

   a. The stand-alone SENSE is never retried for TCP node as it is for a CTC node.
   b. No immediate wait for the completion of the stand-alone SENSE is done within IPW$$LD3. The completion of the stand-alone SENSE is processed via a queued buffer in IPW$$LD1.
   c. A new status byte NCBTPEND is used to issue the message
      1RC6I CONNECTION PENDING FOR NODE ....
      every 12 minutes in case the connection of the two nodes is not yet completely established (the initial and response SIGNON records have not yet been exchanged).

   Thus the stand-alone SENSE results in following socket calls:

   SOCKET To allocate the necessary control blocks to start a new TCP/IP connection.

   GETHOSTBYNAME To get a binary IP-address for the remote node. This socket call is issued only, if a logical hostname has been used in the PNODE macro.

   CONNECT To start a TCP/IP connection to the remote IP-address.

   SEND To send a 33-byte OPEN control record according to the NJE protocol.

   RECV To receive a 33-byte ACK (or NAK) control record according to the NJE protocol.

   CLOSE To stop the TCP/IP connection in case the CONNECT has failed or a 33-byte NAK control record has been received or any other error occurred.

2. READ only request

   The READ only request results in only one socket call:

   RECV To receive NJE data (SOH ENQ)

3. A WRITE only request

   The WRITE only request results in following socket calls:

   SEND To send NJE data (SIGNOFF record)

CANCEL To stop an outstanding receive request. This socket call is issued only if there is a socket call RECV outstanding, but which will be the usual case.

CLOSE To stop the TCP/IP connection.

Once the TCP/IP connection has been closed, the status bit NCBTPS1F is set to signal the PNET Driver (IPW$$LD3) that the TCP/IP connection has been closed and that the final stop activities (remove the NCB out of the NCB chain and release NCB storage) can be performed or that the node can be restarted by the PNET Driver by issuing a new stand-alone SENSE request. The TD Subtask updates the status bits NCBTPS22 or NCBTPS2R, if the TCP/IP error conditions allow a restart.

4. A WRITE/READ request

   This I/O request consists of four CCWs, a SENSE, WRITE, CONTROL and READ CCW. The sense byte for the SENSE CCW is always udated with X'07'. For the CONTROL CCW nothing is done. The WRITE CCW is usually translated to a socket call SEND and the READ CCW to a socket call RECV. The socket calls SEND and RECV are started simultaneously and are completed independently from each other.

   The socket call SEND is issued with the length supplied in the WRITE CCW plus the length of the TCP/IP starting and ending frames. With the currently used product TCP/IP for VSE/ESA the SEND is posted complete only when all data has been sent to the remote node. Theoretically, it may happen that the SEND is posted complete and the returncode signals that just part of the data have been sent to the remote node, in which case the socket call SEND is issued once more with the remaining length of the data to be sent. The socket call SEND uses the same buffer which is used by the WRITE CCW, which means an I/O request with a WRITE CCW can not be flagged complete before the socket call SEND has completed.

   The socket call RECV is issued using a TCP/IP buffer (different from the buffer used in the READ CCW) with the length equal to the buffersize (which is the value of BUFSIZE used in the PNODE macro) plus some extra bytes to contain the TCP/IP starting and ending frames, because one does not know in advance how many bytes the remote node may send to the local node. The returncode of the RECV contains the number of bytes which have been received. Usually one socket call RECV receives all the data sent by one socket call SEND of the remote node. But as the TCP/IP network does not know anything about a logical TCP/IP block of data, it may happen depending on the performance of the network:

   a. that more than one socket call RECV is necessary to receive all data for one TCP/IP block
   b. that more than one TCP/IP block has been received by one socket call RECV

   If the remote node is an RSCS node, it may often occur that more than one socket call RECV is necessary to receive all data for one TCP/IP block, because RSCS issues three socket calls to send one TCP/IP block of data:

   a. The first SEND for the starting frame (TTB, TTR)
   b. The second SEND for the NJE data
   c. The third SEND for the ending frame (TTR-EOB)

   As soon as one block of NJE data has been received, the NJE data is moved from the TCP/IP buffer to the READ CCW buffer.

   All buffers (any SEND or RECV buffers) used for a TCP node are buffers allocated in virtual storage, not in real storage, as the I/O request never ends up in an EXCP REAL request.

Sending Empty Buffers via the TCP/IP Network: If no data has to be sent from one node to another, empty buffers are sent via a CTC line in order to give the other node a chance to start transmission of data. Sending empty buffers is not necessary for TCP nodes, because each node has a socket call RECV outstanding and is therefore ready to receive data from the other node at any time.

Each CTC buffer, even an empty buffer contains:

1. a starting frame (DLE-STX)
2. a block sequence count
3. two function control sequence (FCS) bytes

Processing FCS Bytes The two function control sequence (FCS) bytes control the inbound flow:

1. one bit for each of the eight inbound streams to hold or enable the stream
2. one bit to hold or enable all inbound streams

The FCS bytes are sent via the TCP/IP connection as received via the CTC buffer signaling the remote note to hold or enable the sending of data via its transmitters. If the FCS bytes within the current CTC buffer are different from the FCS bytes within the last CTC buffer, the FCS bytes are sent to the remote node, even via an empty record.

The FCS bytes are set:

1. to hold a stream by the buffer services (IPW$$BS) when the the maximum of queued receive buffers for a stream is reached
2. to hold all streams by the network manager (IPW$$NM), when no buffer can be allocated. The reason could be:
   a. the maximum of receive buffers for the node is allocated
   b. no storage is available

The FCS bytes are set to enable a stream by the buffer services (IPW$$BS) every time a receive buffer is freed. At this point the status for the TCP node is updated to leave the idling state.

Posting an I/O Request Complete: For a CTC node every 1.5 second an I/O request is started, either to send data or an empty buffer. As it is not necessary to send empty buffers via the TCP/IP connection, the CTC I/O request is posted complete only in the following situations.

1. a socket call SEND completed (NJE data has been sent to the remote node)
2. a socket call RECV completed (NJE data has been received from the remote node)
3. both socket calls SEND and RECV completed
4. an 'idling state' must be left (see below)

If a socket call SEND is completed, but the RECV is not complete, the I/O request is posted complete and the READ CCW buffer is updated to contain an empty buffer to acknowledge the sent data of the WRITE CCW.

If a socket call SEND is not complete, but the RECV is complete, the I/O request is not posted complete causing the the CTC write buffer to be freed. Since the CTC write buffer is used as TCP/IP send buffer, the CTC write buffer can not be freed before the socket call SEND is completed.

Idling State: If no data has to be sent to the remote node and the socket call RECV did not complete, the node enters the 'idling state' (NCBTPS3I). This means on the NJE layer the PNET Driver has started an I/O request which remains outstanding, because the TCP/IP layer did not yet complete the I/O request. The idling state is left, if:

1. The outstanding socket call RECV is completed (NJE data has been received from the remote node)

2. The local node has some data to send to the remote node which has been queued by PNET buffer services (IPW$$BS) as the first buffer into one of the to-be-sent-queues (the normal queue or the priority queue). In this case the PNET Driver (IPW$$BS) sets the status bit NCBTPS3L. Next time, when the TD Subtask scans through the NCB-chain, the TD Subtask posts the I/O request complete and the READ CCW buffer is updated to contain an empty buffer. Thus the PNET Driver gets a chance to issue a new I/O request to send some data to the remote node. This happens for example if the PNET line driver sends
   a. a RIF record to start sending data to the remote node
   b. a PGR record to acknowledge the RIF record of the remote node (or NPGR record as negative acknowledgement)
   c. an EOT record to acknowledge the reception of a whole queue entry
   d. messages or commands via the console transmitter
   e. a SIGNOFF record
3. The FCS bytes of the local node have been changed by the PNET Driver and have to be sent to the remote node (see “Processing FCS Bytes” on page 205).

BCB Processing: The block control byte (BCB) is contained within the starting BSC frame for any record, even an empty block. The BCB contains a counter with values from 0 to 15 which starts again with 0 after 15. This counter is used to detect and correct sequence errors. There exists a BCB for the input buffers and another BCB for the output buffers.

Allthough the TCP/IP protocol has probably its own sequence checking, VSE/POWER maintains BCB's within the buffers sent and received via the TCP/IP connection. As usually no empty buffers are sent via the TCP/IP connection, the BCB within the TCP/IP buffers is different from the BCB contained in the current CTC buffers. Therefore VSE/POWER maintains BCB's for the CTC buffer (maintained by the PNET driver) and additional BCB's for the TCP/IP buffers (maintained by the TD Subtask).

The BCB for the TCP/IP send buffer is updated and sent to the remote node without any further processing.

The BCB for the TCP/IP receive buffer is checked for correctness and if invalid, an incorrect BCB is moved into the CTC buffer which causes the PNET driver:

1. to send a "BCB sequence error" record
2. to stop the node
3. to restart the node, if not suppressed by the NR option of the PSTART command.

The TD Subtask does not check the BCB within the CTC send buffer. Since the CTC buffer is used as the TCP/IP send buffer, the BCB within the CTC buffer is temporarily updated with the BCB for TCP/IP. When the CTC I/O is posted complete, the BCB of the CTC send buffer is reset to its original value.

The TD Subtask updates the BCB within the CTC read buffer with the expected value (NCBEBCB) updated by the PNET driver. Only if the TD Subtask detected a BCB error within the BCB of the TCP/IP receive buffer, the BCB of the CTC read buffer is updated with a value which causes a BCB error (processed by the PNET driver, see above).

If a BCB within the TCP/IP receive buffer indicates "reset BCB", the the BCB for TCP/IP send buffer is updated to the recived value, the BCB's of the CTC buffers remain undisturbed.

Control Flows:

Flow of I/O Processing: The Figure 64 on page 207 shows the modules and components involved in processing an I/O request and its corresponding socket calls after a PSTART command has been entered.

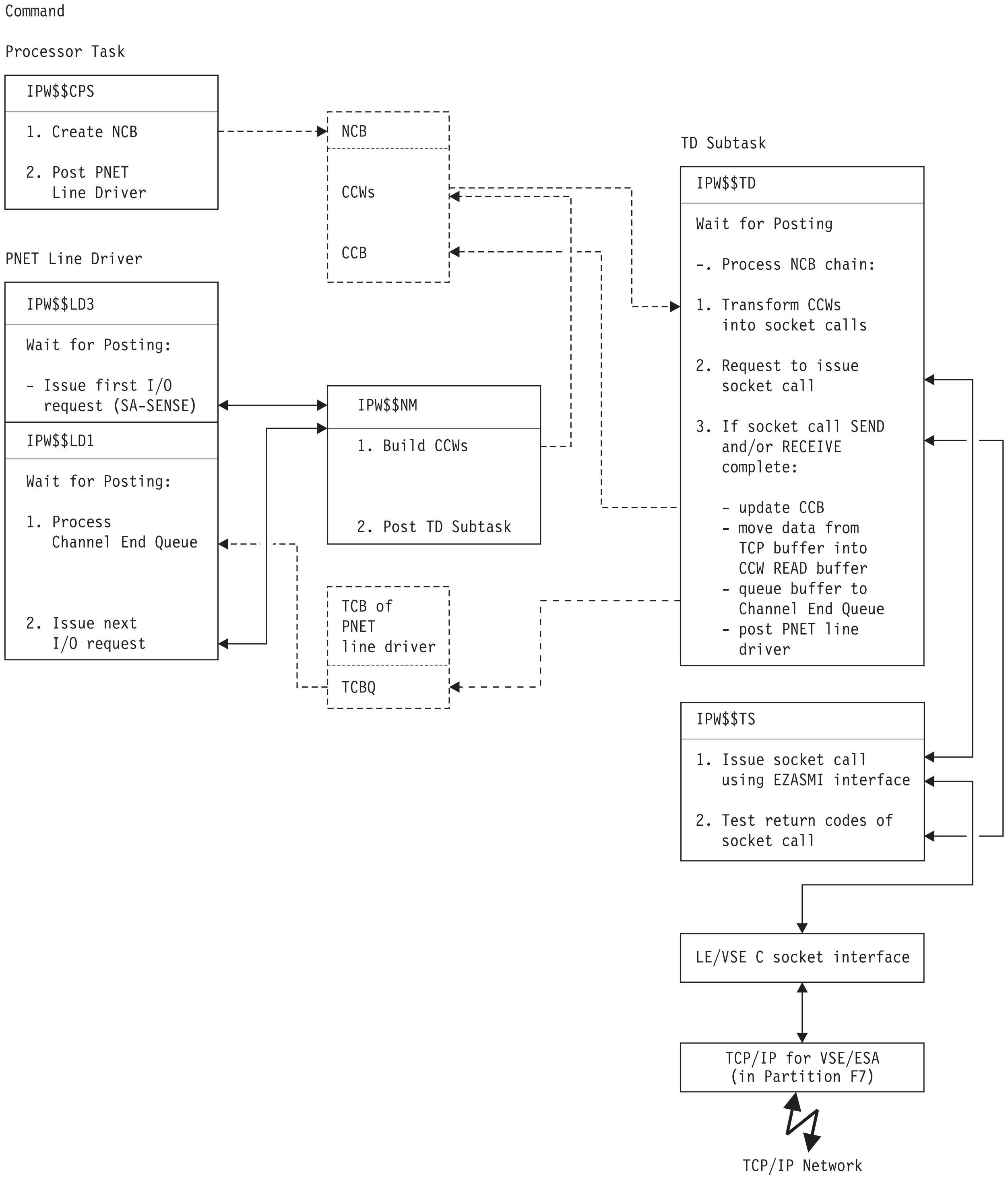


Figure 64. TD-Subtask - Flow of I/O after PSTART

Flow of Processing when Starting First Node: After the TD Subtask has been attached at PNET intialization on both nodes A and B, see Figure 65 on page 208, the subtasks on both nodes prepare themselves to ACCEPT a CONNECT request from the other side. The CONNECT request is triggered by the PSTART PNET,NODEA command entered on NODE B. But the TD Subtask on NODE A rejects the connection, because the corresponding PSTART PNET,NODEB command has not yet been entered at NODE A. The SA-SENSE at node B is not yet posted complete.

```
                 NODE A                                                            NODE B

PLOAD                         TD Subtask                       TD-Subtask                               PLOAD
PNET cmd                      TD-Driver/                       TD-Driver/                               PNET cmd
  *                           TS-Socket                        TS-Socket                                        -----
  |                                                                                                               |
  +----ATTACH-Subtask--->  INITAPI                                                                        *     * * *
                           S0:=SOCKET                                                                     |     * T *
                           BIND(S0) to local port A                                                       |     * C *
                              +                                                                           |     * P *
                           LISTEN(S0)                        INITAPI          <-----ATTACH-Subtask--------+     * * *
                              +                              B0:=SOCKET                                           |
                           S1:=ACCEPT(S0)                    BIND(B0) to local port B                           * * *
                           wait A-ECB                          +                                                * L *
                              +                              LISTEN(B0)                                         * I *
                              +                                +                                                * S *
                              +                              ACCEPT(B0)                                         * T *
                              +                              wait A-ECB                                         * E *
                              +                                +                              PNET              * N *
                              +                                +                            LD-Driver           * * *
                              +                                +                                      enter cmd --|--
                              +                                +                              + PSTART PNET,NODEA |
                              +                                +                              |                   |
                              +                                + C1:=SOCKET   <---------SA-SENSE cmd-code       * * *
                              +                                + CONNECT(C1) to A                               * T *
                              * :<-----post-A-ECB-for-S1------------                          .                 * C *
                                :-------post-C-ECB-----------> + wait C-ECB                   .                 * P *
                                RECEIVE(S1)                    +  :                           .                 * * *
                                wait R-ECB(S1)                 +  :                           .                   |
                                :          <---post-R-ECB(S1)  + SEND(C1) OPEN-CTL-rec        .                 * * *
                                verify OPEN-CTL versus TCP-    +  :                           .                 * I *
                                PNODE entries of local NDT     +  :                           .                 * N *
                                and started TCP-nodes (NCB)    + RECEIVE(C1)                  .                 * N *
          >>>> assume 'no PSTART yet'                          + wait R-ECB(C1)               .                 * I *
                                CLOSE(S1)-post-R-ECB(C1)       +  :                           .                 * T *
                                          with data-len=0----> +  :                           .                 * * *
               refresh ... S2:=ACCEPT(S0)                      + RC=-1,ERRNO=1124 or recv-len=0                   |
                           wait A-ECB                          +  :                           . => 1RC6I CONNECT  |
1RTHI NODE B AWAITING  <===== +                                +                                         PENDING  |
      CONNECTION              +                                +                                                  |
```

Figure 65. TD Subtask - Startup and First PSTART Command

Flow of Processing when Starting Second Node: With the first connection attempt rejected in Figure 65, and NODE B in 'CONNECTION PENDING FOR NODE A' state, the scenario continues with Figure 66 on page 209, when finally the missing PSTART PNET,NODEB command is entered on NODE A side.

```
                  NODE A                                                            NODE B

        PNET                   TD Subtask                       TD-Subtask                     PNET
      LD-Driver                TD-Driver/                       TD-Driver/                   LD-Driver
                               TS-Socket                        TS-Socket

           (for entry, see precedeing figure 'Startup TD Subtask - Incoming CONNECT Request')
                              +                                 +                                                  |
                 (wait Accept-ECB outstanding)        (wait Accept-ECB outstanding)                                |
                              +                                 +                               .                * * *
 enter cmd                    +                                 +                               .                * T *
 PSTART PNET,NODEB            +                                 +                               .                * C *
       |                      +                                 +                               .                * P *
    SA-SENSE cmd-code-------> + C2:=SOCKET                      +                               .                * * *
         .                    + CONNECT(C2) to B -post-A-ECB B1--> B1:=ACCEPT(B0)               .                  |
         .                    + wait C-ECB <---post-C-ECB------    :                            .                * * *
         .                    +  :                                RECEIVE(B1)                   .                * I *
         .                    + SEND(C2) OPEN-CTL-rec ---->       wait R-ECB(B1)                .                * N *
         .                    + RECEIVE (C2)                       :                            .                * I *
         .                    + wait R-ECB(C2)                     :                            .                * T *
         .                    +  :      <-ACK-CTL-rec------       SEND(B1)                      .                * * *
         .                    +  :             refresh .... ACCEPT(B0)                          .                  |
         |<---'07'----for CTL-+  :                           wait A-ECB                         .                  |
         .                    +  :                              +  : ----for-no-cmd--'00'-->    |                  |
         .                    +  :                              +  :                         CONTROL               |
         .                    +  :                              +  :                         READ                  |
  >>>>>  . >>>>>>>>>>> all subsequent DATA exchange with TTB|TTR|DATA|TTR-EOB format only  <<<< . <<<<<<<<<<<<<    |
         .                    +  :                              +  :                            .                  |
       *****                  +  :                              +  :                            .               -----
       SENSE                  +  :                              +                                                  |
       WRITE SOH ENQ ------->-+--:                              + RECEIVE (B1)                  .                  |
         .                    + 'C2' socket related             + wait R-ECB                    .                  |
         .                    + SEND -SOH-ENQ-buffer---->       +  :                            .                  |
         .                    +  :                              +  :   --------SOH ENQ ----- >  |                * * *
         .                    +  :                              +  :                          *****              * C *
         .                    + RECEIVE                         +  :                          SENSE              * T *
         .                    + wait R-ECB                      +  :   <-------DLE ACK0------ WRITE              * C *
       CONTROL                +  :      <---DLE-ACK0-from---    + SEND                          .                * * *
       READ <------DLE ACK0---+- :                              +  :                            .                  |
       *****                  +  :                              +  :                            .                * * *
  >>> from now on the DATA in TTB|TTR|DATA|.... has NJE-CTC buffer format: DLE|STX|BCB|FCS|FCS|RCB....  <<<<<<   * I *
         .                    +  :                              + RECEIVE                       .                * N *
       SENSE                  +  :                              + wait R-ECB                  CONTROL            * I *
       WRITE 'I' signon rec->-+--:                              +  :                            .                * T *
         .                    + SEND 'I'-signon-rec----->       +  :                            .                * * *
         .                    +  :                              +  : --------'I'-signon rec->-READ                 |
         .                    +  :                              +  :                          ***=> MSG 1RB3I SIGNED ON
         .                    +  :                              +  :                          SENSE                |
         .                    + RECEIVE                         +  :                            .                  |
         .                    + wait R-ECB                      +  :  <------'J' signon resp- WRITE                |
       CONTROL                +  :      <-'J'-signon response- +-SEND                           .                  |
       READ <-'J'-signon resp-+--:                              +  :                            .                  |
1RB3I<=*****                  +  :                              +  :                            .                  |
SIGNED ON                     +  :                              +  :                            .                  |
   ----------------- N O D E S -------- A R E -------- S I G N E D ---------- O N -------------------------------------
```

Figure 66. TD Subtask - Second PSTART Command and Signon Complete

Flow of Processing the Idling State

The idling state is entered, if both nodes issued a CTC I/O request to send an empty buffer. Since empty buffers are not sent via the TCP/IP connection, both CTC I/O requests are not posted complete, and the nodes are set into "idling state".

If node A wants to send a record (containing a message for example) to node B, the record is queued as first buffer in the to-be-sent-queue and the status in the NCB is updated to leave the idling state. Next time, the TD Subtask is posted due to its own timer interval, the TD Subtask processes the new status of the NCB. Since no data has been received via the TCP/IP communication, the TD Subtask completes the CTC I/O and passes back an empty buffer. Thereafter the PNET driver starts a CTC I/O request to send the record to node B. The TD Subtask processes the CTC I/O request and issues a socket call SEND to send the record via the TCP/IP communication. The TD Subtask completes the CTC I/O request immediately and passes back an empty buffer, in order to let the PNET driver send some more data.

```
                   NODE A                                                            NODE B

         PNET                   TD Subtask                       TD-Subtask                     PNET
       LD-Driver                TD-Driver/                       TD-Driver/                   LD-Driver
                                TS-Socket                        TS-Socket

                               +  :                              +  :                          *****
                               +  :                              +  :                          SENSE
                               +  :                              +  : <--- empty buffer -----  WRITE
                               +  :                              +  no SEND for empty bfr |--    .
                               +  :                              +  RECEIVE                      .
                               +  :                              +  wait R-ECB                 wait for I/O completion
        *****                  +  :                              +  :                            .
        SENSE                  +  :                              +  :                            .
        WRITE -- empty buffer --->no SEND for empty bfr |        +  :                            .
          .                    +                                 +  :                            .
 wait for I/O completion       +  still wait R-ECB               +  :                            .
          .                    +  :                              +  :                            .
          .                    +  :                              +  :                            .
          .             --------- Idling state: both CTC I/O not yet complete -------            .
          .                    +  :                              +  :                            .
          .                    +  :                              +  :                            .
 'message to be sent'==|       +  :                              +  :                            .
          .            |========> update NCB status              +  :                            .
        CONTROL                +  :                              +  :                            .
        READ <--'empty buffer'----complete I/O                   +  :                            .
        *****                  +  :                              +  :                            .
        SENSE                  +  :                              +  :                            .
        WRITE --- message ------->:                              +  :                            .
          .                    +  SEND --- message --------|     +  :                            .
          .                    +  RECEIVE                  |     +  :                            .
        CONTROL                +  wait R-ECB               |     +  :                            .
        READ <-- empty buffer ----complete I/O (may be ..  |     +  :                            .
        *****                  +  : ..more data to send)   |     +  :                            .
        SENSE                  +  :                        |     +  :                            .
        WRITE -- empty buffer --->no SEND for empty bfr|   |     +  :                            .
          .                    +  :                        |     +  :                            .
 wait for I/O completion       +  still wait R-ECB         |     +  :                            .
          .                    +  :                        |---> +  R-ECB posted by TCP/IP       .
          .                    +  :                              +  :                          CONTROL
          .                    +  :                              +  : --- message -----------> READ
          .                    +  :                              +  :                          *****
          .                    +  :                              +  :                          SENSE
          .                    +  :                              +  : <--- empty buffer -----  WRITE
          .                    +  :                              +  no SEND for empty bfr|       .
          .                    +  :                              +  RECEIVE                      .
          .                    +  :                              +  wait R-ECB                 wait for I/O completion
          .                    +  :                              +  :                            .
          .             --------- Idling state: both CTC I/O not yet complete -------            .
```

Figure 67. TD Subtask - Processing the Idling State

TCP/IP Driver Subtask Mainline (IPW$$TD):

Overview: The module IPW$$TD consists of following parts:

TDCS Initialization after being attached

TDMAIN Loops forever untill a stopcode is set. Following processing occurs within this loop:

TDTERM	Terminates TCP/IP interface if necessary
TDINITAP	Initializes TCP/IP interface if necessary
TDPASSOK	Processes as server in "passive mode"
TDNBLPIN	Loops through the NCB-chain to process all TCP-nodes (in 'active mode')
TDPOSLDR	Posts PNET Driver if necessary
TDDETACH	Detach itself (the TD Subtask) if necessary
TDTIMSET	Issues timer interval using TQE within subtask control block (TDCB)
TDWAITDS	Issues VSE/ESA wait macro If posted, starts loop from the beginning

TDSBATDY Tidy-up routine in case of abnormal termination, called by IPW$$AT.

TDCS - Initialization

The part TDCS performs following initialization steps after being attached:

1. Initialize base register
2. Initialize register for save area
3. Initialize register for save area used by macro IPW$IDM
4. Drop protection key zero in order to run in parallel mode, if multiprocessor support is available
5. Initialize areas for the timer services within TDCB
6. Initialize TCP/IP workareas within TDCB
   a. Clear TCP/IP workarea for passive mode
   b. Clear workarea within module IPW$$TD
   c. Clear local IP-address which is updated later on by the socket call GETHOSTID
   d. Set trace option on for socket calls issued during initialization
   e. Set request 'initialize interface to TCP/IP'
7. Update task identifiers (name and VSE/ESA subtask identifier) in TDCB
8. Anchor address of tidy-up routine TDSBATDY in TDCB, which is used for abnormal termination in IPW$$AT
9. Initialize timer services (use macro IPW$TTM with operand STXIT=YES)

TDMAIN - Mainloop

Overview: The part TDMAIN contains the mainloop of the TD Subtask, which is left only if some stop conditions cause the detach of the TD Subtask. In case of an abnormal termination this loop is not entered, but the tidy-up routine TDSBATDY is entered directly from IPW$$AT. At the end of the mainloop, the VSE/ESA WAIT macro is issued using as ECB the field TDCBECB within the TDCB. TDCBECB is posted due to the events described above (see “Posting Events” on page 200). When the TD Subtask is posted, the mainloop is started again at label TDMAIN. At the beginning of the mainloop following steps are performed:

1. Clear the ECB (TDCBECB)
2. Call timer services to process expired timer intervals (use macro IPW$TTM with operand PROCESS=YES)

Following steps are part of the mainloop:

TDTERM: The interface to TCP/IP is terminated by issuing the socket call TERMAPI. It is terminated according to the stop condition: if the interface has to be terminated immediately, the socket call TERMAPI is issued at once. In all other cases, the TERMAPI is postponed untill all TCP nodes (TDCBNONU must contain 0) have been stopped and the passive connection has been stopped. In case the initialization (socket call INITAPI) has failed, no socket call TERMAPI is issued. In any case, any outstanding message of type A (anchored in TDCBMSGD) is deleted via macro IPW$GTO using the operand DOM. Thereafter a terminating message, either 1RT8I or 1RTSI, is issued via macro IPW$GTO with the operand MSG.

TDINITAP: The interface to TCP/IP is initialized by issuing the socket call INITAPI, but only if no stop condition is set. In addition the status "issue startup-message" (TDCBS1SM) is set. The startup-message 1RT7I is issued once for each initial socket call (SOCKET, GETHOSTID, BIND, LISTEN). Thereafter the initialzation process for the passive mode is entered by entering the routine TDINIGSC directly.

TDPASSOK: The part TDPASSOK contains the code for the TD Subtask to run as a server in passive mode. Following steps are performed:

1. If the interface to TCP/IP is not available or the passive mode has been stopped, passive mode processing is bypassed and processing continues with searching through the NCB-chain.
2. If the passive mode has to be stopped, processing continues with closing the passive connection (TDPASCLS)
3. If the passive mode has been started (a CONNECT request from a remote node has been received), but the initial contact (the exchange of the OPEN and ACK or NAK control records) did not complete in time, the processing continues with closing the passive connection (TDPASTIO)
4. If a special event (TDNTPS4P) has to be processed, processing continues with the routine for this event (TDNTPWPO). At this point only one event may happen:
   a. When receiving a CONNECT, the NCB chain had to be scanned for matching definitions. For this purpose the PNCB had to be locked. If the locking of the PNCB had failed, the status (TDNTPS4P) is updated to reenter the routine (TDPASVLF) to lock the PNCB.
5. In all other cases processing continues with the routine to issue a socket call (TDSBSOCK).

When routine TDSBSOCK is entered, register 1 points to the parameter area (TDNTPDS) which contains all necessary information to issue a socket call, namely:

1. the type of socket call (TDNTPSC)
2. the return addresses (TDNTPS00, TDNTPS04, TDNTPS08, TDNTPS0C) for the routine TDSBSOCK according to the returncode (0,4,8,12) set by the service routine (IPW$$TS)

The parameters are set by the routines initializing the socket call before entering the routine TDPASSOK. This means the routine TDPASSOK is entered directly by subroutines to issuing a socket call and thereafter, once the socket call has been issued, by the mainline to check for the completion of the socket call.

Following socket calls are issued in passive mode

1. The first sequence of socket calls is issued to prepare the passive mode.

   The return address (TDNTPS00) for the returncode 0 is set in each routine before the socket call is issued. The return addresses (TDNTPS04, TDNTPS08, TDNTPS0C) for the error conditions are set just once and have been initialized in routine TDINITAP before issueing the socket call INITAPI. TDINITAP passes control to routine TDINIGSC to start the initial sequence of socket calls.

   Following routines are involved in preparing passive mode processing:

   TDINIGSC Issue the socket call SOCKET to allocate the necessary control blocks for the passive mode. The parameter area is initialized with the port number (TDNSCBPT) out of the NDT and the family type (TDNSCBFM) is set to 2. Returned is the socket descriptor (TDNSCRC), a number which is saved into TDNSCSOD, because it must be referenced by all following socket calls. In addition the status "TCP/IP interface available" (TDCBS1IA) is set, because the socket call SOCKET was the first, which has been passed through all the TCP/IP layers. If this socket calls fails, the type A message 1RTJA is issued, which will be deleted if the interface has been established successfully or the TD Subtask is stopped. If the returncode makes a retry meaningful, the socket call is retried every 20 seconds.

TDINIGIP Issue the socket call GETHOSTID to get the binary IP-address of the local node and update TDCBSCIR with the readable IP-address.

TDINIBID Issue the socket call BIND to link the socket with a port number.

TDINILIS Issue the socket call LISTEN to prepare the socket for the next socket call ACCEPT. The status "issue startup-message" (TDCBS1SM) is reset and any outstanding message of type A (anchored in TDCBMSGD) is deleted via macro IPW$GTO using the operand DOM.

2. The next sequence of socket calls is issued to process incoming CONNECT request from remote nodes and complete the initial contact (see “Processing as Server and Client” on page 201) The return addresses (TDNTPS00, TDNTPS04, TDNTPS08, TDNTPS0C) are set in the routines to the appropriate values.

   Following routines are involved in passive mode processing:

   TDPASACC Issue the socket call ACCEPT to get posted whenever a CONNECT request is received. The posted ACCEPT returns a new socket descriptor (TDNSCSOD), because new control blocks have been allocated by the TCP/IP layers when the CONNECT has been received. This new socket descriptor is now used for all following socket calls. The socket descriptor used for the socket call ACCEPT has been saved into TDSVSCSD in routine TDINIGSC and is used for all new ACCEPT socket calls. The received binary IP-address of the remote node is translated into readable format and put into TDNTPIPC.

   In addition a timer interval is set for 5 minutes. Within this time limit the initial contact must be completed, otherwise the local node stops the connection by issuing a socket call CLOSE and message 1RTGI. This avoids that a hanging connection prevents that other connections can not be established, because one CONNECT request after the other is processed sequentially.

   TDPASGHN Issue the socket call GETHOSTBYADDR to get a logical hostname. The previously posted ACCEPT returned just a binary IP-address, but no logical host name. When checking the network definition table (NDT) both IP-addresses, the binary IP-address and the logical hostname, are used to check whether a node has been defined for one of these addresses.

   If no matching node has been found in the NDT, the message 1RT3I is issued. If a logical hostname was found for the received binary IP-address, the message 1RTBI is issued in addition.

   TDPASRCV Issue the socket call RECV to receive an OPEN control record.

   Following error messages are issued:

   1RT4I The control record was not of type OPEN. The connection is stopped by issuing a socket call CLOSE. No NAK control record is sent.

   1RT5I The control record contained invalid information, which might be the FROM node-id or FROM IP-address or the TO node-id or TO IP-address. The connection is stopped by issuing a socket call CLOSE after a NAK control record with RC=1 has been sent.

   1RTEI The binary IP-address was not used for the node-id found in the NDT according to the binary IP-address received by the CONNECT request. The new connection is stopped by sending a NAK control record with RC=1 and issuing a socket call CLOSE.

1RTBI The logical hostname found due to the binary IP-address of the CONNECT request was not used for the node-id found in the NDT. The new connection is stopped by sending a NAK control record with RC=1 and issuing a socket call CLOSE.

1RTHI A PSTART command has not yet been issued for the node-id. The new connection is stopped by issuing a socket call CLOSE without sending a NAK control record.

1RTVI A PSTART command has already been issued for the node-id. One of the following situation has occurred:

a. The TD Subtask is just starting a connection for this node-id in active mode. Therefore the new connection, started by the passive mode, is stopped by sending a NAK control record with RC=3 and issuing a socket call CLOSE. The status of the connection starting in active mode is not changed at all and continues its normal flow.
b. The TCP/IP connection for the node-id has been established some time ago successfully. Therefore the new connection, started by the passive mode, is stopped by sending a NAK control record with RC=2 and issuing a socket call CLOSE. The already active connection is stopped as well by setting the stopcode to line-error.

1RTFI The received control record is displayed in hexadecimal format. This message is displayed in addition to one of the previous messages (1RT4I or 1RT5I).

TDPASSND Issue the socket call SEND to send the ACK or NAK control record. If an ACK control record has been sent, the status bytes within the NCB are updated to continue processing for this node (in active mode) using the TCP/IP connection established in passive mode. If a NAK control record has been sent, the passive connection is closed by issuing a socket call CLOSE. In both cases the workarea for the passive connection within the TDCB is cleared to process new incoming CONNECT requests. Processing continues in routine TDPASACC with issuing a socket call ACCEPT.

TDNBLPIN: The part TDNBLPIN contains the code for the TD Subtask to run as a client in active mode. Following steps are performed (for more details see “TDNBLPIN - Loop through NCB-Chain (Details)” on page 215):

TDNBLPIN Init search through NCB chain

TDNBLPNX For each TCP node in NCB chain:

TDNBCLOS Close connection if necessary

TDNBINIT Start initial contact if necessary

TDNBWAIT Process next NCB if processing for current NCB is postponed

TDNBIOST Start processing of CTC I/O request: translate CCW's

TDNBIOSN Process SENSE CCW

TDNBIOWR Process WRITE CCW

TDNBIOCL Process CONTROL CCW

TDNBIORD Process READ CCW

TDNBFCSW Init wait timer interval if FCS signals "hold stream(s)"

TDNBIDLG Leave idling state if necessary

TDNBIOZ0 Complete I/O if necessary

TDNBTIME Init timer interval for this NCB if necessary

TDPOSLDR: If the status "post PNET Driver" (TDCBA3PL) is set, the ECB (TCEB) of the PNET Driver is updated and the main-ECB (PAEB within the CAT) of the VSE/POWER maintask is posted by using the VSE/ESA macro POST. TDCBA3PL has been set, if an I/O request for a node has been completed and an input buffer has been queued to the PNET Driver TCB in routine TDNBIOZ0.

TDDETACH: If the status "detach task" (TDCBA1DT) has been set in routine TDTERM after the socket call TERMAPI has been issued, the TD Subtask is now detached by issuing the VSE/ESA macro DETACH. The TD Subtask can not be detached immediately in routine TDTERM, because in case of immediate termination of the TD Subtask, the status bytes for the TCP nodes have to be set first by entering routine TDNBLPIN. Before detaching the TD Subtask, the timer service is called by using the macro IPW$TTM with the operand CANCEL to cancel an outstanding timer interval using the TQE within the TDCB.

TDTIMSET: First the timer service is called by using the macro IPW$TTM with the operand CANCEL to cancel an outstanding timer interval using the TQE within the TDCB. Thereafter the timer service is called by using the macro IPW$TTM with the operand TIME to issue a new timer interval using the TQE within the TDCB.

TDWAITDS: If the status "omit WAIT" (TDCBS2NW) is not set, the VSE/ESA macro WAIT is issued using the ECB (TDCBECB) within the TDCB. TDCBS1NW is set in case the TCP/IP interface is no longer available in error routines TDINITII and TDPASSTI. In these cases a WAIT is not necessary, the mainloop is directly reentered at label TDMAIN to terminate and detach the TD Subtask.

TDNBLPIN - Loop through NCB-Chain (Details)

Following steps are processed for each NCB of the NCB chain.

TDNBLPIN: Before searching through the NCB chain for a TCP node, the PNCB is locked using the test and set (TS) instruction. If the locking is successful, the lockword is updated with "TCP" to identify the TD Subtask as owner of the PNCB. If the locking is unsuccessful, the TD Subtask waits for 1 second using the macro IPW$TTM with the operand "REACTIVATE" before retrying to lock the PNCB.

The PNCB is unlocked, if the end of the PNCB chain is reached or if a TCP node is found for which the TCP/IP connection has not yet been closed (NCBTPS1F).

TDNBCLOS: Before a socket call CLOSE is issued, a socket call CANCEL is issued for an outstanding SEND and/or RECV socket call. Depending on the status bytes within the NCB (NCBTPSTx):

1. A message is issued explaining the reason why the the TCP connection has been closed
2. If a SIGNOFF record has been received, processing continues at TDNBTIME to terminate the TCP/IP connection
3. If TCP/IP connection to be restarted:
   a. Clear TCP/IP workarea (starting at NCBTPDS)
   b. Processing continues at TDNBTIME to init timer interval before restarting
4. In all other cases, processing continues to complete I/O (TDNBIOZ0)

TDNBINIT: The routine TDNBINGS (for more details see “TDNBINGS - Establish Initial Contact (Details)” on page 218) to establish the initial contact is entered:

1. If the initial contact is not yet complete
2. And if no wait is issued
3. And if a wait has expired in case a wait has been iussed earlier
4. And if a connect request from the remote node is not processed by the passive mode
5. And if the TCP/IP interface is available

TDNBWAIT: If a timer interval has been set and is not yet expired, processing continues with the next NCB. If a timer interval has been set and has expired:
if received or transmitted FCS bytes of CTC buffers still hold a stream, status "complete I/O" is set to give the PNET driver a chance to update FCS bytes (see “TDNBFCSW” on page 217).

TDNBIOST: If status is "start I/O processing", initialize address of last CCW processed and issue trace message containing CCW-chain, if console trace is started.

TDNBIOSN - Loop through CCW-Chain

Start processing of CCW-chain, if TCP/IP connection not closed:

1. If stand-alone SENSE CCW, continue to finish CCW processing (TDNBIOLC).
2. Otherwise update sense byte with op-code CONTROL and continue processing with next CCW (TDNBIONC).

TDNBIOWR: If WRITE CCW:

1. If socket call SEND started, continue with checking for completion of SEND socket call.
2. If CTC I/O already once processed, continue processing next CCW (meaningful if CTC buffer contained empty buffer)
3. If signon complete, FCS bytes did not change and CTC buffer contains empty buffer, omit socket call SEND and continue processing next CCW (TDNBIONC)
4. Update BCB in TCP/IP send buffer (as the CTC write buffer is part of the TCP/IP send buffer, the BCB of the CTC buffer is first saved)
5. FCS bytes of CTC write buffer remain unchanged in TCP/IP send buffer
6. Update TTB, starting TTR and TTR-EOB in TCP/IP send buffer
7. Update address and length of data and return addresses according to the return code of the socket call
8. Call subroutine to issue send request
9. If socket call SEND completed:
   a. If not all bytes sent, update address and length of data and continue processing with subroutine call to issue send request
   b. Restore BCB in CTC write buffer
   c. If SOH-ENQ or ACK sent, do not update status "complete CTC I/O", but wait till SIGNON record has been received
   d. If SIGNOFF record sent, continue processing with closing the TCP/IP connection (TDNBCLOS)
   e. In all other cases update status "complete CTC I/O" and continue processing next CCW (TDNBIONC)

TDNBIOCL: If CONTROL CCW, continue processing next CCW (TDNBIONC).

TDNBIORD: If READ CCW:

1. If CTC I/O not yet processed once and FCS bytes hold at least one strean and console trace is started, issue trace mesage with FCS bytes of CTC write buffer.
2. If FCS bytes hold all streams, omit socket call RECV (if FCS bytes hold just one stream, we trust remote node and assume the received buffer will never be for the suspended stream. RSCS processes the FCS bytes too late and sends buffer even for a suspended stream, but the PNET driver ignores this protocol violation and continues processing without any error indication)
3. If socket call RECV started, continue with checking for completion of RECV socket call.
4. If no data left over from last socket call RECV, continue with starting socket call RECV
5. If received data in TCP/IP receive buffer contains all bytes for a CTC read buffer (without TTB and TTR):
   a. Move data from TCP/IP receive buffer to CTC read buffer

   b. If BCB of TCP/IP receive buffer is invalid, update BCB of CTC read buffer to an invalid value which forces a BCB sequence error issued by the PNET driver (node is stopped and restarted)
   c. If console trace is started, issue message displaying TTR and first 12 bytes of CTC read buffer
   d. If socket call SEND started and not yet complete, omit setting of status "complete CTC I/O"
   e. Continue processing next CCW (TDNBIONC).
6. Start processing of socket call RECV
   a. If any bytes left after having moved some data from TCP/IP receive buffer into CTC read buffer, move left data to begin of TCP/IP receive buffer and display data via trace message, if console trace started
   b. Update address and length of data and return addresses according to the return code of the socket call
   c. Call subroutine to issue receive request
   d. If socket call RECV completed:
      1) If no bytes received, continue processing with closing the TCP/IP connection
      2) Continue processing above with checking, if all bytes for a CTC read buffer (without TTB and TTR) have been received
   e. If socket call RECV should be retried (return code = 4 of socket call):
      1) If signon not yet complete, continue processing with next CCW (and wait till all data received)
      2) If CTC I/O to be completed (send socket call completed), continue processing with next CCW (and do not wait till receive complete)
      3) If no data received and if no send socket call outstanding, set status "idling"
      4) Continue processing next CCW (TDNBIONC).

TDNBIONC - processing next CCW: If CCW is not last CCW, update address of last CCW processed (NCBLCCW) and continue to process the CCW (TDNBIOSN)

If CCW is last CCW, update status "I/O once processed".

TDNBFCSW: If signon is complete (FCS bytes are contained within CTC buffers) and received or transmitted FCS bytes hold at least one stream and the CTC I/O can not be completed, continue processing to initiate wait for a timer interval of 20 seconds to suspend processing for this node (see “TDNBWAIT” on page 216).

TDNBIDLG: If status is "leave idling state" and "idling", set status "complete I/O".

TDNBIOZ0: To complete the CTC I/O, following steps are performed:

1. If status is not "line busy" and not "close line", issue IDUMP macro and message 1RTLI.
2. Update address of last processed CCW (NCBLCCW) to point after last processed CCW
3. Update CCB status with "channel and device end"
4. If no data received via TCP/IP connection, build empty buffer
5. Update residual count in CCB
6. If console trace started, issue trace message to display CCB, address of last processed CCW, and some status bytes
7. Queue buffer to channel end queue of PNET driver via compare and swap instruction
8. If SIGNOFF record received, continue processing with closing the TCP/IP connection (TDNBCLOS)
9. If TCP/IP connection closed and to be restarted, init timer interval for restart
10. If TCP/IP connection closed and not to be restarted, set status "terminate connection"

TDNBTIME: If status is "terminate connection":

1. Cancel any outstanding TQE for this NCB and update status "TCP/IP connection finished" (NCBTPS1F, used by PNET driver to start final node clean up).
2. If no SIGNOFF record received nor sent, update status with TCP/IP error.
3. Continue processing next NCB (TDNBNEXT).

If timer interval to set, initiate timer interval using value (in NCBTPTIV) previously set and update status "waiting for expiration of timer interval" (NCBTPS4W).

TDNBINGS - Establish Initial Contact (Details)

The TD Subtask performs following steps to establish the initial contact as a client in active mode:

TDNBINGS Issue the socket call SOCKET to allocate the necessary control blocks for a connection in active mode.

TDNBINGH If a logical IP address has been specified for the NCB, issue the socket call GETHOSTBYNAME to get a binary IP address.

TDNBINCO Issue the socket call CONNECT to send a connect request to the remote node.

TDNBINSR If CONNECT completed successfully, issue the socket call SEND to send an OPEN control record to the remote node.

TDNBINRR If SEND completed successfully, issue the socket call RECV to receive an ACK control record from the remote node.

If no ACK nor NAK control record has been received, issue messages 1RTDI and 1RTFI, and continue processing to close the TCP/IP connection.

If the ACK or NAK control record contains incorrect values, issue messages 1RT5I and 1RTFI, and continue processing to close the TCP/IP connection.

If a NAK control record has been received, issue message 1RT6I and continue processing to close the TCP/IP connection. If a NAK control record with RC=3 has been received, update status "restart TCP/IP connection" (NCBTPS1R).

If an ACK control record has been received:

1. Set status "initial contact complete"
2. Set status "complete I/O"
3. Update sense bytes with CCW op-code CONTROL
4. Continue processing with process CTC I/O (TDNBIOST).

IPW$$TD - Tidy-up Routine - TDSBATDY: This routine is called by module IPW$$AT in case of an abnormal termination of the TD Subtask. Following steps are performed:

1. The timer service of the VSE/ESA supervisor is terminated by issuing the VSE/ESA macro STXIT with the operand IT. This terminates any timer service established by one of the TCP/IP layers which have been called when a socket call is issued.
2. The anchor point (TDCBTQEA) for the timer service is cleared.
3. The following resources are unlocked by clearing the lockword:

   TIBLCK The trace information block which might have been locked because trace entries have been written into the storage trace area

   CAAB The asynchronous service anchor block which might have been locked because the filled up storage trace area had to be written to disk

4. The main-ECB (PAEB within the CAT) of the VSE/POWER maintask is posted by using the VSE/ESA macro POST in order to resume processing of any task waiting for one of the above resources.
5. Any outstanding message of type A (anchored in TDCBMSGD) is deleted via macro IPW$GTO using the operand DOM.
6. In order to stop all TCP nodes the PNCB is locked. If the PNCB can not be locked, a wait for a second is issued by using the macro IPW$TTM with the operand REACTIVATE to be posted after 1 second
7. In order to stop all TCP nodes as fast as possible the following information is set

a. NCBTPS4C - Socket call CLOSE has been issued
b. NCBTPS1F - TCP/IP connection is finished
c. NCBF1BY - No I/O request outstanding
d. NCBTTCL - Line error
e. NCBLNSR - Close line, used by the activity-process of the PNET Driver (IPW$$LD3)
f. The post bit within the ECB (TCEB) of the PNET Driver is set.

8. The PNCB is unlocked and the main-ECB (PAEB within the CAT) of the VSE/POWER maintask is posted by using the VSE/ESA macro POST.
9. If the TCP/IP interface was available, the socket call TERMAPI is issued.
10. Reset status "interface available" and status "interface once available", other status information is reset when entering the initialization process (TDCS).
11. Issue message 1RT8I TCP/IP: INTERFACE NOT AVAILABLE
12. Reload saved registers and return to caller (IPW$$AT).

TCP/IP Driver Subtask Services Interface Macros (IPW$$TS): In order to separate subtask service functions from the TCP/IP driver subtask mainline, the following services are provided (see Appendix C, “VSE/POWER Internal Macros” on page 747 for more details):

1. EZASMI Socketcall Support
2. Message Support
3. Timer Interval Interrupt Supprt
4. EZASMI Socketcall Error Checking Support

In addition, the Service Support module has its own internal tracing.

Subtask EZASMI Socketcall Support (IPW$ITP Macro): Using the IPW$ITP macro, the subtask may invoke the EZASMI API and check for error conditions afterwards.

IPW$ITP PARMS=socketcall: Using the IPW$ITP macro, the subtask may invoke the EZASMI API for the following socketcalls:

- ACCEPT
- BIND
- CANCEL
- CLOSE
- CONNECT
- INITAPI
- GETHOSTID
- GETHOSTBYADDR
- GETHOSTBYNAME
- LISTEN
- RECEIVE
- SEND
- SOCKET
- TERMAPI

The EZASMI interface is invoked in 31-bit mode.

Internally the IPW$$TS module will invoke the IPW$ITP CKRC=YES macro to check the EZASMI socketcall for any immediate error return.

IPW$ITP CKRC=YES: This macro is called internally in the IPW$$TS module to check for immediately returned EZASMI API access errors, and also by the IPW$$TD module to check for errors following EZASMI API ECB posting.

Subtask Message Support (IPW$GTO Macro): Since a VSE/POWER subtask cannot use the messaging support available to the maintask, the following functions are provided with their own access macro.

IPW$GTO MSG=msgid: This access macro allows the caller to specify the message equate "msgid" of a message defined by the IPW$GMM macro in the IPW$$MM module. The message will be issued in the same way as for the maintask, using the WTO macro and providing message substitution and message squeezing via the IPW$$MX module.

IPW$GTO MSG=TRACE: This access macro allows the caller to issue a PNET Driver Subtask trace message (1RTTI).

IPW$GTO DOM=(R1): This access macro allows the caller to delete a console message issued previously by the IPW$GTO MSG= macro.

Subtask Timer Interval Interrupt Support (IPW$TTM Macro): The are various support access macros:

IPW$TTM STXIT=YES: This access macro initializes the VSE Timer STXIT interface for the SETIME macro used for the other support macros.

IPW$TTM TIME=(Rx),TQE=: This access macro allows the caller to indicate a timer interval in tenths of a second following which an ECB is posted in the indicated TQE element and the Driver Subtask is also posted.

IPW$TTM CANCEL=YES,TQE=: The caller indicates that a previous IPW$TTM TIME= request is to be cancelled.

IPW$TTM PROCESS=YES: This access macro is called by the Driver Subtask following posting.

IPW$TTM WAIT=(Rx): This access macro allows the Driver Subtask to indicate it wishes to go into a wait state until it is posted by either the expiration of a SETIME interval request for the WAIT= interval (in tenths of a second), or by any other event which may occur sooner, with the register Rx containing the interval value.

IPW$TTM WAIT=(Rx,REACTIVATE): This access macro allows the Driver Subtask to indicate it wishes to go into a wait state as for the IPW$TTM WAIT(Rx) macro, and additionally the macros IPW$TTM STXIT=YES and IPW$TTM PROCESS=YES are called immediately following.

Subtask Support Internal Trace: The support module IPW$$TS has its own internal tracing area. Each module entry and exit is recorded in the trace area with an eye catcher and register contents. At the end of the IPW$$TS module, beginning at the eye catcher "LAST ENTRY =" lies the trace area. The layout area is:

- eye catcher "LAST ENTRY="
- address of the last trace entry that was recorded (4 bytes)
- eye catcher "LAST BRANCH="
- address to which the module last exited
- (80 byte entries) with the layout:
  - eye catcher describing the entry (16 bytes)
  - contents of the registers 0 to 15
- eye catcher "$$STBUF END"

PNET SSL Interface to TCP/IP

Establishing the Interface to TCP/IP SSL: At PNET initialization by IPW$$IN, when the internal PLOAD PNET command is launched, or at any later time, when an operator requests (re)-loading of the Network Definition Table by the PLOAD command, module IPW$$CLD checks the new NDT for at least one valid SSL node and invokes then the PSTART TCPSSL command. The IPW$$CS command processor accepts this startup attempt for the SD Subtask (representing the TCP/IP interface) only by a VSE/POWER internal request and does the following:

1. Lock/Unlock the PNCB to record the temporary PSTART command task in TDCBATCB (of SDCB) as a serial resource, which intends to attach the SD Subtask. When another PSTART TCPSSL task is pending, new attempts are rejected.
2. Check the VSE ATTACH/DETACH communication TDCBSECB (within SDCB) of the SD Subtask for subtask still down or already alive. If the subtask has been started previously and no stop code is pending, then terminate the PSTART task. If subtask is active, but stop codes are pending in TDCBSTA1 or TDCBACT1, then the PSTART task checks periodically until the SD Subtask has terminated in order to attempt a new attach request.
3. Use VSE ATTACH to give control to the SD Subtask for module IPW$$SD.
4. Finally lock/unlock the PNCB to clear the PSTART task pointer in TDCBATCB (of SDCB) and terminate this command processor.

The SD Subtask is attached with the following processing attributes, which outline the subtask's embedding within the surrounding VSE/POWER functions and services. The SD Subtask

- calculates its entry point into module IPW$$SD after
  1. the VSE/POWER storage descriptor,
  2. the VSE subtask save area,
  3. the Subtask-id 'IPW$$SD', and
  4. the Subtask abnormal termination save area
- uses VSE/POWER maintask's ABEXIT in IPW$$AT
- drops protection key 0 (established during attach) in order to run in parallel mode
- provides local save area for requesting Idumps in flight using VSE/POWER's IPW$IDM support for a VSE Subtask
- provides local Message Control Block for requesting IPW$$MX message modification for IPW$$MM defined messages
- uses macro IPW$GTS to request message support from TCP/IP Service Module IPW$$SS
- uses macro IPW$TTS to request STXIT timer interval support from TCP/IP Service Module IPW$$SS
- uses macro IPW$ITS to request EZASMI Socket calls from TCP/IP Service Module IPW$$SS
- uses VSE macro WAIT TDCBECB (in SDCB) to be activated by the PNET Driver from the VSE/POWER CTC layer
- uses VSE macro POST PAEB (with Line Driver task ecb posted additionally) to activate the VSE/POWER CTC layer from the TCP/IP interface layer
- uses Test-and-Set instruction to lock the PNCB for sharing the NCB chain with the Line Driver task, the TD Subtask, and the PSTART PNET command processor task - marking PNCB lockword with 'SSL', when owned by SD Subtask
- uses Compare-and-Swap instruction to add an 'I/O completed' CTC input buffer to the Line Driver Channel End Queue for sharing this resource with the Line Driver task, the TD Subtask and the I/O Supervisor Task
- cannot use macro IPW$GTE to reserve an entry of the telecommunication trace area or to call trace area dumping, instead ...
- uses Test-and-Set instruction to lock the Trace Information Block (TIB) for sharing line trace entries with the PNET Line Driver task, the TD Subtask, and the RJE Line Manager task - marking TIB lockword with 'SSL', when owned by SD Subtask

- uses Test-and-Set instruction to lock the Asynchronous Service Anchor Block (ASAB) when sharing trace area dumping with the PNET Line Driver task, the TD Subtask, and the RJE Line Manager task - marking ASAB lockword with 'SSL', when owned by SD Subtask
- calls macro IPW$IAS to invoke the Dump Subtask of IPW$$AS for trace area dumping. Selected parts of this module have been made sensitive on 'being called by VSE/POWER task or by SD Subtask'

Controlling the TCP/IP SSL Interface: Once the SD Subtask has been attached (see message 1RVMI) it remains active, even when another NDT is loaded lateron without any remote SSL node. The 'active' state can be interrogated by either command

- STATUS F1 (assuming VSE/POWER in F1), presenting the 'IPW$$SD' subtask of F1
- PINQUIRE NODE=local-node, presenting '1R56I TCP SSL: ...' information.

The SD Subtask is only terminated by external request

- at normal VSE/POWER session termination time through the PEND command, which sets TDCBACT1 flags in the SDCB. Then, after all SSL nodes have been stopped, the SD Subtask requests DETACH. When however all (non core) VSE/POWER tasks have terminated and the termination processor IPW$$T1 still finds the SD Subtask communication TDCBSECB with 'alive' indication, there is safety code, so that the termination task requests DETACH for the SD Subtask.
- during a VSE/POWER session through
  - the PSTOP TCPSSL (EOJ) command, which informs the SD Subtask for termination processing as if PEND were given.
  - the PSTOP TCPSSL,FORCE command. This format is only intended for halting the SD Subtask abruptly, in case it seems to 'hang'. Hence the IPW$$CP stop command processor requests subtask cancellation by the TREADY call (with cancel code X'08' = 'due to PSTOP), which leads to AB-Exit processing in IPW$$AT.
  - an abnormal Subtask termination (e.g. program check) leading to AB-Exit processing in IPW$$AT, which has as well standard VSE subtask tidy-up steps (message 1Q2CI and Idump) as a subtask type specific step. For the SD Subtask the TCP/IP related tidy-up routine is called, which is located in module IPW$$SD, and which terminates all SSL node processing and the interface to TCP/IP for VSE. Upon return to IPW$$AT the SD Subtask finally requests DETACH.

TCP/SSL Driver Subtask (SD Subtask)

Overview: The TCP/SSL Driver subtask or SD Subtask processes as a VSE/ESA subtask using the modules IPW$$SD and IPW$$SS. IPW$$SD is the main routine and calls IPW$$SS for following purposes:

1. Issue a socket call
2. Test the returncode of a socket call
3. Issue a message
4. Use timer services:
   a. Initialize timer services
   b. Set up a timer interval
   c. Process expired timer intervals
   d. Cancel a timer interval
   e. Wait a bit

In addition following modules, which are used mainly by the VSE/POWER maintask, are used by the SD Subtask as well:

IPW$$AT	Process abnormal termination
IPW$$AS	Write storage trace entries to dump libraries

The SD Subtask starts its processing after it has been attached by the VSE/POWER maintask when a network definition table is loaded with at least one SSL node.

The SD Subtask ends its processing

1. In normal situations due to any PSTOP or PEND command by detaching itself using the VSE/ESA macro DETACH
2. In abnormal situations by returning to IPW$$AT after some cleanup processing

The main purpose of the SD Subtask is to process requests concerning a SSL node:

1. Translate an I/O request consisting of several CCW's (built according to the CTC protocol) into socket calls and call the TCP/IP layer by using the macro EZASMI
2. Process the completion of a socket call
3. Pass the received data of a socket call into the buffer of a read CCW and queue this buffer to the channel-end-queue which is processed by the PNET Driver
4. Process any normal or abnormal stop condition

TCP/SSL and TCP/IP Driver Subtasks (TD and SD Subtasks): The TCP/SSL Driver subtask or SD Subtask processes as a VSE/ESA subtask all nodes using a TCP/IP connection together with SSL feature, whereas the TCP/IP Driver subtask or TD Subtask processes as a VSE/ESA subtask all nodes using a TCP/IP connection without SSL feature. The SD Subtask uses the modules IPW$$SD and IPW$$SS, the TD Subtask uses the modules IPW$$TD and IPW$$TS. Since the processing of the SD Subtask and the TD Subtask is very similar, the modules IPW$$SD and IPW$$SS are a copy of the modules IPW$$TD and IPW$$TS with following adaptions:

1. After the initial contact for a node has been established, the SSL feature is used to send data to or receive data from a remote node. Following socket calls of the SSL feature are used:

   GSKINIT
   to initialize once the SSL for VSE/ESA environment for the VSE/POWER partition

   GSKUNINIT
   to remove the current overall settings for the SSL environment

   GSKGETCIPHINF
   to request cipher related information for SSL for VSE/ESA

GSKSSOCINIT
to initialize the data areas necessary for SSL for VSE/ESA to initiate or accept a secure socket connection

GSKSSOCREAD
to receive data on a secure socket connection

GSKSSOCWRITE
to send data on a secure socket connection

GSKSSOCCLOSE
to close a secure socket connection and free all SSL for VSE/ESA resources for that connection

GSKGETDNBYLAB
to get the label for a key in a key database file

GSKFREEMEM
to free memory that was passed to the application on a previous call to an SSL function.

2. Since the socket calls GSKSSOCREAD and GSKSSOCWRITE are processed synchronously, whereas the socket calls RECV and WRITE of the TD Subtask are processed asynchronously, the SD subtask uses following socket calls:

   IOCTL to set nonblocking mode for a subsequent socket call CONNECT
   IOCTL to set blocking mode after a previously issued socket call CONNECT
   SELECT to test if a socket is ready for a subsequent ACCEPT, READ or GSKSSOCREAD, SEND or GSKSSOCWRITE.

3. Since all socket calls are processed synchronously, the socket call CANCEL to terminate an outstanding socket call (for example WRITE or RECV) is not used by the SD Subtask.

To establish a connection between two nodes using the SSL feature, some socket calls are used which are the same as for a a connection between two nodes not using the SSL feature. Once the 'initial contact' for a connection has been established, the SSL feature is used to exchange data (jobs, list or punch output, messages and commands, network control records). The 'initial contact' is complete, if the socket call CONNECT completed successfully and the client sent an 'OPEN control record' and received an 'ACK control record'.

Before issuing any SSL socket call destined for a connection, the overall SSL environment must be established by using the GSKINIT which must be issued just once, whereas the GSKSSOCINIT must be issued once for each connection.

At the end of processing the overall SSL environment is freed by using the GSKUNINIT.

In order to initialize the SSL feature for a socket, the GSKGETDNBYLAB is used, the output of which is used as input for the GSKSSOCINIT. To initialize the appropriate ciphers, the GSKGETCIPHINF is used, the output of which is also used as input for the GSKSSOCINIT.

```
         CLIENT                                   SERVER

                                         initapi<-----------------------

                                           gsk_initialize <-------------

                                         socket <-----------------------
                                         bind   <-----------------------
                                         listen <-----------------------
                                         accept <-----------------------
------------------->  initapi

------> gsk_initialize

------------------->  socket
------------------->  connect

------------------->  send 'OPEN'
                                         receive 'OPEN' <---------------
                                         send    'ACK ' <---------------
------------------->  recv 'ACK '

------> gsk_get_dn_by_label                gsk_get_dn_by_label  <-------
------> gsk_get_cipher_info                gsk_get_cipher_info  <-------

------> gsk_secure_soc_init                gsk_secure_soc_init  <-------

------> gsk_secure_soc_write
                                           gsk_secure_soc_read  <-------
                                           gsk_secure_soc_write <-------
------> gsk_secure_soc_read

------> gsk_secure_soc_close               gsk_secure_soc_close <-------

------------------->   close             close <------------------------

------> gsk_uninitialize
------------------->  termapi
                                           gsk_uninitialize <-----------
                                         termapi<-----------------------
```

Figure 68. Socket Calls Used

The GSKSSOCINIT requires as input parameters pointers to

1. a routine SKWRITE which is called to send data
2. a routine SKREAD which is called to receive data

These routines are called by

1. the GSKSSOCINIT during the establishing of a secured connection to exchange information concerning the algorithms used for encryption and certification.
2. the GSKSSOCWRRITE after the data to be sent has been encrypted
3. the GSKSSOCREAD to receive data which is thereafter decrypted

These routines issue the 'normal' synchronous send and receive socket calls. PNET SSL does not specify any routines, but uses the default send and receive routines distributed by VSE/ESA.

```
                             CLIENT                                  SERVER


                                                              accept <---------------------
                    ------------------->  socket
                    ------------------->  connect
                    ------------------->  send 'OPEN'
                                                              receive 'OPEN' <-------------
                                                              send    'ACK ' <-------------
                    ------------------->  recv 'ACK '

                    ------> gsk_secure_soc_init               gsk_secure_soc_init  <-------
                                (SKWRITE,SKREAD)                  (SKWRITE,SKREAD)
                                        -------> write
                                                          read  <------
                                                          write <------
                                        -------> read
                                        -------> write
                                                          read  <------
                                                          write <------
                                        -------> read

                                                  ..............


                    ------> gsk_secure_soc_write
                                        -------> write
                                                              gsk_secure_soc_read  <-------
                                                          read  <------

                                                              gsk_secure_soc_write <-------
                                                          write <------
                    ------> gsk_secure_soc_read
                                        -------> read


                                                  ..............


                    ------> gsk_secure_soc_close              gsk_secure_soc_close <-------

                    ------------------->   close            close <------------------------
```

Figure 69. Synchronous Socket Calls Read and Write Used Implicitly

During the initiailization of the secure connection via the GSKSSOCINIT usually more than one write and read socket call are issued by both nodes.

Interfaces and Operation Layers: The SD Subtask translates an I/O request (which has been built by the PNET Driver) into socket calls. To issue the socket call, the SD Subtask communicates with the product TCP/IP for VSE/ESA parts of which run in its own partition (usually F7). Figure 70 shows the three layers involved in this communication.

```
                           NODE A                                            NODE B

      I                   II            III     (IV)    III              II                 I

   PNET SSL                           TCP/IP           TCP/IP                            PNET SSL
  Line Driver  <---> SD Subtask <--->  part.--INTERNET--part.  <---> SD Subtask <--->  Line Driver

 |---P O W E R ---PARTITION----|    |TCP-PART|        |TCP-PART|    |---P O W E R----PARTITION-----|
```

Figure 70. SD Subtask - Three Operation Layers for PNET SSL support.

The SD Subtask, however, does not communicate directly with TCP/IP for VSE. Instead, it communicates with the LE/VSE C socket interface, which directly communicates with TCP/IP for VSE. To address the LE/VSE C socket interface, VSE/POWER uses an application interface (EZASMI macro) which has been introduced with VSE/ESA 2.5. Some routines of these three components are loaded into the VSE/POWER partition, some into the SVA.

The communication between the VSE/POWER partition and the TCP/IP partition (F7 in the system setup distributed by VSE/ESA) is done by routines of the product TCP/IP for VSE/ESA which uses the VSE/ESA XPCC interface.

Posting Events: The SD Subtask gets posted by the

1. Command processor (IPW$$CPS) to start its processing (when being attached)
2. VSE/ESA Supervisor due to an expired timer interval set up by the SD Subtask itself
3. PNET Driver (IPW$$NM) to translate an I/O request
4. PNET buffer services (IPW$$BS) to complete an outstanding I/O request because an output buffer has been put into an empty buffer queue. This can be caused by one of the following tasks:
   a. by a transmitter task (even by a console transmitter)
   b. by a receiver task (queuing a PGR, NPGR, or EOT record)
   c. by the PNET Driver task (queuing a network control record, for example a SIGNON or SIGNOFF record)
5. Command processor (IPW$$CP) to terminate its own processing (due to a PSTOP TCPSSL command).

Note however:

1. If a PSTOP PNET,nodeid command without the FORCE operand has been entered, the connection to a node is terminated properly by sending a SIGNOFF record to the remote node, which means the SD Subtask gets posted by the PNET Driver (IPW$$NM) due to a new I/O request.
2. If the command PSTOP PNET,nodeid,FORCE (with the FORCE operand) has been entered, the PNET Driver is posted but not the SD Subtask. The connection is terminated unproperly, no SIGNOFF record is sent to the remote node. The SD Subtask processes the immediate stop request, when posted due to its own timer services.
3. If the command PSTOP TCPSSL,FORCE (with the FORCE operand) has been entered, the SD Subtask is cancelled for entering the abnormal termination routine IPW$$AT.
4. If the PSTOP TCPSSL command without the FORCE operand has been entered, the SD Subtask waits till all nodes are stopped:
   a. If a node is signed-on, the node must be stopped explicitly by issuing a PSTOP PNET,nodeid command.
   b. If a node is not signed-on, the connection is terminated unproperly, no SIGNOFF record is sent to the remote node.

Addressing Mode: The SD Subtask runs usually in 24-bit addressing mode. Running within module IPW$$SD, the SD Subtask runs for a few instruction in 31-bit addressing mode when addressing data received by following socket calls:

GETHOSTBYADDR receiving a logical hostname according to the received binary IP-address via a socket call CONNECT.

GETHOSTBYNAME receiving a binary IP-address according to the logical hostname specified in the network definition table (NDT)

Running within module IPW$$SS, the SD Subtask runs in 31-bit addressing mode when issuing a socket call.

Processing as Server and Client: Within a TCP/IP network, an application may run as a client or server. The client is the application which issues a CONNECT request, the server is the application which issues the socket calls BIND, LISTEN and ACCEPT and processes a CONNECT request of another client. The client is said to run in 'active mode', whereas the server is said to run in 'passive mode'.
The SD Subtask processes as a server and as a client. Whenever a PSTART command for a node has been entered, the SD Subtask issues a CONNECT request and acts as a client. If the remote node does not answer or rejects the CONNECT request, the SD Subtask suspends its active mode for a while (usually 2 minutes). During this time a CONNECT request can be received from the remote node, and the SD Subtask acts then as a server. In order to process incoming CONNECT requests from remote nodes, the SD Subtask issues an ACCEPT request at the beginning of its processing. Whenever the ACCEPT request signals an incoming CONNECT request, the SD Subtask processes this request and thereafter issues a new ACCEPT request for more CONNECT requests. As soon as a connection to a remote node has been established successfully, all connections are internally flagged as processing in 'active mode'.

Processing Socket Calls Synchronously: Since the socket calls GSKSSOCREAD and GSKSSOCWRITE used for the SSL feature are processed synchronously, the SD Subtask has to process all socket calls synchronously, including the socket calls CONNECT and ACCEPT. To prevent any blocking of the SD Subtask processing, the socket call SELECT is issued to test if a subsequent socket call ACCEPT, READ or WRITE completes immediately.

Before issuing a socket call CONNECT the socket call IOCTL is issued to set the connection to non-blocking mode. Thus the socket call CONNECT completes immediately and a subsequent socket call SELECT will indicate whether the CONNECT completes successfully or not. After the socket call CONNECT the socket call IOCTL is issued once more to reset the connection back to blocking mode.

Processing the Socket Call SELECT: The socket call SELECT is used to avoid that the SD Subtask hangs when issuing a synchronous socket call which can not complete immediately. Following socket calls are used:

SELECT using the "read array"
to test if a socket is ready for a subsequent ACCEPT, RECV or GSKSSOCREAD. A mask in the read array is set on whenever a new socket has been allocated (either by a socket call SOCKET or ACCEPT). A mask in the read array is set off whenever a socket has been deallocated (by a socket call CLOSE). The socket call is issued in the mainline after the TD Subtask has been posted ready by the VSE/AF supervisor.

SELECT using the "write and exception array"
to test if a socket is ready for a subsequent SEND or GSKSSOCWRITE. The socket call is issued in the subroutine before the SEND or GSKSSOCWRITE is issued. A mask in the write and exception array is set on right before and set off right after the SEND or GSKSSOCWRITE is issued.

To keep the two modules IPW$$SD and IPW$$TD as similar as possible the socket call SELECT is issued in following routines:

TDWAITDS is part of the mainline where the socket call SELECT is issued after the SD Subtask has been posted ready by the VSE/AF supervisor to test if a socket is ready for a subsequent ACCEPT, RECV or GSKSSOCREAD. The socket call SELECT uses only the read array. A mask in the read array is set on whenever a new socket has been allocated (either by a socket call SOCKET or ACCEPT). A mask in the read array is set off whenever a socket has been deallocated (by a socket call CLOSE). If the socket call SELECT completes unsuccessfully, the read array is cleared to zero and a mask is set on again as described below.

TDSBSOCK is the subroutine which is called to issue a socket request.

1. Whenever a socket call ACCEPT, RECV or GSKSSOCREAD is issued, a test is done, if the mask for the currently processed connection is set on. If the mask is off (because the

SELECT at label TDWAITDS failed), the mask is set on for this connection and a socket call SELECT with the updated read array is issued. If the mask is on, the socket call SELECT is omitted to improve the overall performance of the system and not to seize the system by the SD Subtask.

2. Whenever a socket call SEND or GSKSSOCWRITE is issued, the mask is set on for this connection in the write and exception array and a socket call SELECT is issued. After the socket call SEND or GSKSSOCWRITE has been issued, the mask is set off for this connection in the write and exception array.

Processing the 'Initial Contact': The usage of a TCP/IP network as a physical layer for a logical NJE network has been first implemented by RSCS. Hence VSE/POWER implemented the same rules:

1. A connection is established according to the CTC protocol, which means at the beginning of the connection the BSC characters SOH-ENQ and DLE-ACK0 are exchanged.

2. All data exchanged according to the CTC protocol are blocked into a TCP/IP block using the following structure:

TTB	8 bytes describing a block of NJE data
TTR	4 bytes describing a record of NJE data
--	n bytes containing NJE data
TTR-EOB	4 bytes describing the end of a block of NJE data

   At this point, however, one TCP/IP block contains only one record of NJE data

3. Before starting to exchange data according to the CTC protocol, an 'initial contact' is established, namely two control records, an OPEN and an ACK or NAK control record, are exchanged to verify that the two nodes adhere to the NJE protocol. The OPEN control record is exchanged first, whereas the ACK or NAK control record is sent as response to the OPEN control record. The ACK is sent as a positive acknowledgement to continue with the connection, whereas the NAK is sent as a negative acknowledgement to stop the connection. All control records contain the following 33 bytes:

   a. 8 bytes describing the type of the control record
   b. 8 bytes describing the FROM NJE nodename
   c. 4 bytes describing the FROM IP-address
   d. 8 bytes describing the TO NJE nodename
   e. 4 bytes describing the TO IP-address
   f. 1 byte describing a return-code, which is used only for a NAK control record

   To differentiate between SSL and TCP nodes, the type of the OPEN control record is initialized with "OPEN SSL".

   If an SSL node receives an OPEN control record of type "OPEN   " (without the characters SSL), the SSL node sends a NAK control record of type "NAK SSL", to indicate to the remote node, that the SSL feature must be used. If A NAK control record of type "NAK   " and return-code 4 (indicating SSL feature not used) would be sent to RSCS which does not yet support the SSL feature, RSCS would try forever to start the connection.

Details about the TCP/IP frames and control records are described in the macro IPW$DTP, starting at the lables TCPTTB, TCPTTR and TCPCTRL.

During the initial contact no socket calls of the SSL feature are used due to the following reasons:

1. If a TCP node (not using the SSL feature) tries to connect to an SSL node (using SSL feature), the TCP node would receive encrypted data which would not result in meaningful error messages.
2. If the passive connection (due to received CONNECT request from a remote started node) and the active connection (due to PSTART command on the local node) are processing the same remote

node, the socket call GSKSSOCINIT would result in a contention situation which can not be solved due to the synchronous socket call GSKSSOCINIT.

After the initial contact has been completed successfully, following socket calls of the SSL feature are issued:

1. GSKGETCIPHINF to get the cipher related information due to the value specified in the operand ENCRYPT of the PNODE macro.
2. GSKSSOCINIT to initialize the SSL feature for the connection.
3. GSKSSOCREAD to receive data.
4. GSKSSOCWRITE to send data.

Processing I/O Requests: The I/O requests processed by the SD Subtask are built according to the CTC protocol by the PNET Driver (IPW$$NM). Instead of issuing a START I/O request, the PNET Driver updates the status 'I/O request to be processed' (NCBTPS3S) and posts the SD Subtask. When the SD Subtask gets control, it loops through the NCB-chain and finds the I/O request to be processed (NCBTPS3S). To flag the I/O request 'complete', the SD Subtask updates the CCB and queues a buffer to the channel-end-queue anchored in the PNET Driver TCB (TCBQ) using Compare-and-swap (like the Channel-End-Appendage routine for BSC- and CTC-nodes and TD Subtask). The SD Subtask updates the CCB always with channel and device end, which means the PNET Driver will never issue any special I/O request for recovery purposes. If any error occurs, the SD Subtask informs the PNET Driver (IPW$$LD1) by setting NCBTPS1E, which causes the node to be stopped on the NJE layer.

The PNET Driver issues only the following CTC I/O requests for a SSL node:

1. Stand-alone SENSE CCW

   The I/O request consists of one SENSE CCW only. This request is issued only as the first request when starting the connection for a node. It is used to synchronize the I/O requests with the remote node. The input for a SENSE CCW is one byte, the command code pending on the remote node. The SD Subtask returns only two different command codes:

   a. X'07' (CTC Control), if the remote node did not yet issue an I/O request. The PNET Driver issues as response to this sense byte an I/O request containing four CCWs, a SENSE, WRITE, CONTROL and READ CCW. The WRITE CCW sends an SOH-ENQ to the remote node and the READ CCW should receive an DLE-ACK0 from the remote node.
   b. X'00', if the remote node issued already an I/O request. The PNET Driver issues as response an I/O request containing two CCWs, a CONTROL and READ CCW. The READ CCW should receive an DLE-ACK0 from the remote node.

   Although for a CTC line other values than these two may be returned for a SENSE CCW, the SD Subtask restricts itself to these values which are sufficient to handle the two different events, whether the remote node issued already an I/O request or not.

2. A READ only request

   The I/O request consists of two CCWs, a CONTROL and READ CCW. This I/O request is issued only as a response to X'00' received by a stand-alone SENSE CCW. The READ CCW should receive an SOH-ENQ from the remote node.

3. A WRITE only request

   The I/O request consists of two CCWs, a SENSE and WRITE CCW. This I/O request is issued only when the connection to the remote node has to be stopped according to the NJE protocol: a SIGNOFF record is sent to the remote node without waiting for any response.

4. A WRITE/READ request

The I/O request consists of four CCWs, a SENSE, WRITE, CONTROL and READ CCW. This I/O request is issued in all cases except the three cases described above.

The SD Subtask performs following actions for the above described I/O requests:

1. Stand-alone SENSE CCW

   This I/O request is not completed before the initial contact (see “Processing as Server and Client” on page 228) has been done successfully, which means the OPEN and ACK control records have been exchanged. Once a PSTART command has been entered, the SD Subtask issues a CONNECT request to start a TCP/IP connection in active mode. If the CONNECT request fails or a NAK control record is received instead of an ACK control record, the SD Subtask retries the the CONNECT request, usually every 2 minutes, or completes the initial contact via the passive mode in case the remote node started the connection with a CONNECT request.
   This means that the PNET Driver (IPW$$LD3) has been changed:

   a. The stand-alone SENSE is never retried for SSL node as it is for a CTC node.
   b. No immediate wait for the completion of the stand-alone SENSE is done within IPW$$LD3. The completion of the stand-alone SENSE is processed via a queued buffer in IPW$$LD1.
   c. A new status byte NCBTPEND is used to issue the message
      1RC6I CONNECTION PENDING FOR NODE ....
      every 12 minutes in case the connection of the two nodes is not yet completely established (the initial and response SIGNON records have not yet been exchanged).

   Thus the stand-alone SENSE results in following socket calls:

   SOCKET to allocate the necessary control blocks to start a new TCP/IP connection.

   GETHOSTBYNAME to get a binary IP-address for the remote node. This socket call is issued only, if a logical hostname has been used in the PNODE macro.

   IOCTL to set nonblocking mode for the connection to avoid that the subsequent CONNECT blocks the connection in case the remote node does not reply to the CONNECT request

   CONNECT to start a TCP/IP connection to the remote IP-address.

   IOCTL to set blocking mode again for the connection which is needed for all subsequent socket calls (SEND, RECV, GSKSSOCWRITE,..).

   SELECT using the write and exception array to test if the socket is ready for a subsequent SEND to avoid that the SD Subtask hangs in case the SEND can not be completed immediately.

   SEND to send a 33-byte OPEN control record according to the NJE protocol.

   SELECT using the read array to test if the socket is ready for a subsequent RECV to avoid that the SD Subtasks hangs in case the remote node did not sent any data.

   RECV to receive a 33-byte ACK (or NAK) control record according to the NJE protocol.

   CLOSE to stop the TCP/IP connection in case the CONNECT has failed or a 33-byte NAK control record has been received or any other error occurred.

2. READ only request

   The READ only request results in following socket calls:

   SELECT using the read array to test if the socket is ready for a subsequent GSKSSOCREAD to avoid that the SD Subtasks hangs in case the remote node did not sent any data.

   GSKSSOCREAD To receive NJE data (SOH ENQ)

3. A WRITE only request

   The WRITE only request results in following socket calls:

   SELECT — using the write and exception array to test if the socket is ready for a subsequent GSKSSOCWRITE to avoid that the SD Subtask hangs in case the GSKSSOCWRITE can not be completed immediately.

   GSKSSOCWRITE — to send NJE data (SIGNOFF record)

   CLOSE — to stop the TCP/IP connection.

   Once the TCP/IP connection has been closed, the status bit NCBTPS1F is set to signal the PNET Driver (IPW$$LD3) that the TCP/IP connection has been closed and that the final stop activities (remove the NCB out of the NCB chain and release NCB storage) can be performed or that the node can be restarted by the PNET Driver by issuing a new stand-alone SENSE request. The SD Subtask updates the status bits NCBTPS22 or NCBTPS2R, if the TCP/IP error conditions allow a restart.

4. A WRITE/READ request

   This I/O request consists of four CCWs, a SENSE, WRITE, CONTROL and READ CCW. The sense byte for the SENSE CCW is always udated with X'07'. For the CONTROL CCW nothing is done. The WRITE CCW is usually translated to a socket call GSKSSOCWRITE and the READ CCW to a socket call GSKSSOCREAD. The socket calls GSKSSOCWRITE and GSKSSOCREAD are started only, if a previously issued socket call SELECT has indicated that the socket is ready for the socket call.

   The socket call GSKSSOCWRITE is issued with the length supplied in the WRITE CCW plus the length of the TCP/IP starting and ending frames. With the currently used product TCP/IP for VSE/ESA the GSKSSOCWRITE is posted complete only when all data has been sent to the remote node. Theoretically, it may happen that the GSKSSOCWRITE is posted complete and the returncode signals that just part of the data have been sent to the remote node, in which case the socket call GSKSSOCWRITE is issued once more with the remaining length of the data to be sent. The socket call GSKSSOCWRITE uses the same buffer which is used by the WRITE CCW, which means an I/O request with a WRITE CCW can not be flagged complete before the socket call GSKSSOCWRITE has completed.

   The socket call GSKSSOCREAD is issued using a TCP/IP buffer (different from the buffer used in the READ CCW) with the length equal to the buffersize (which is the value of BUFSIZE used in the PNODE macro) plus some extra bytes to contain the TCP/IP starting and ending frames, because one does not know in advance how many bytes the remote node may send to the local node. The return code of the GSKSSOCREAD contains the number of bytes which have been received. Usually one socket call GSKSSOCREAD receives all the data sent by one socket call GSKSSOCWRITE of the remote node. But as the TCP/IP network does not know anything about a logical TCP/IP block of data, it may (depending on the performance of the network) happen:

   a. that more than one socket call GSKSSOCREAD is necessary to receive all data for one TCP/IP block
   b. that more than one TCP/IP block has been received by one socket call GSKSSOCREAD

   If the buffersize is larger than 16K, the current implementation of TCP/IP for VSE/ESA requires more than one socket call GSKSSOCREAD to receive all data for one TCP/IP block, because the GSKSSOCWRITE sends at most 16K in one buffer and therefore splits data larger than 16K.

   As soon as one block of NJE data has been received, the NJE data is moved from the TCP/IP buffer to the READ CCW buffer.

   All send and receive buffers used for an SSL node are buffers allocated in virtual storage, not in SETPFIX LIMIT storage, as the I/O request never ends up in an EXCP REAL request.

Sending Empty Buffers via the TCP/IP Network: If no data has to be sent from one node to another, empty buffers are sent via a CTC line in order to give the other node a chance to start transmission of data. Sending empty buffers is not necessary for SSL nodes, because each node issues always a SELECT using the read array to test if the socket is ready to receive data from the other node at any time.

Each CTC buffer, even an empty buffer contains:

1. a starting frame (DLE-STX)
2. a block sequence count
3. two function control sequence (FCS) bytes

Processing FCS Bytes The two function control sequence (FCS) bytes control the inbound flow:

1. one bit for each of the eight inbound streams to hold or enable the stream
2. one bit to hold or enable all inbound streams

The FCS bytes are sent via the TCP/IP connection as received via the CTC buffer signaling the remote note to hold or enable the sending of data via its transmitters. If the FCS bytes within the current CTC buffer are different from the FCS bytes within the last CTC buffer, the FCS bytes are sent to the remote node, even via an empty record.

The FCS bytes are set:

1. to hold a stream by the buffer services (IPW$$BS) when the the maximum of queued receive buffers for a stream is reached
2. to hold all streams by the network manager (IPW$$NM), when no buffer can be allocated. The reason could be:
   a. the maximum of receive buffers for the node is allocated
   b. no storage is available

The FCS bytes are set to enable a stream by the buffer services (IPW$$BS) every time a receive buffer is freed. At this point the status for the SSL node is updated to leave the idling state.

Posting an I/O Request Complete: For a CTC node every 1.5 second an I/O request is started, either to send data or an empty buffer. As it is not necessary to send empty buffers via the TCP/IP connection, the CTC I/O request is posted complete only in the following situations.

1. a socket call GSKSSOCWRITE completed (NJE data has been sent to the remote node)
2. a socket call GSKSSOCREAD completed (NJE data has been received from the remote node)
3. both socket calls GSKSSOCWRITE and GSKSSOCREAD completed
4. an 'idling state' must be left (see below)

If a socket call GSKSSOCWRITE is completed, but the GSKSSOCREAD is not complete, the I/O request is posted complete and the READ CCW buffer is updated to contain an empty buffer to acknowledge the sent data of the WRITE CCW.

If a socket call GSKSSOCWRITE is not complete, but the GSKSSOCREAD is complete, the I/O request is not posted complete causing the the CTC write buffer to be freed. Since the CTC write buffer is used as TCP/IP send buffer, the CTC write buffer can not be freed before the socket call GSKSSOCWRITE is completed.

Idling State: If no data has to be sent to the remote node and the socket call GSKSSOCREAD did not complete, the node enters the 'idling state' (NCBTPS3I). This means on the NJE layer the PNET Driver has started an I/O request which remains outstanding, because the TCP/IP layer did not yet complete the I/O request. The idling state is left, if:

1. The outstanding socket call GSKSSOCREAD is completed (NJE data has been received from the remote node)
2. The local node has some data to send to the remote node which has been queued by PNET buffer services (IPW$$BS) as the first buffer into one of the to-be-sent-queues (the normal queue or the priority queue). In this case the PNET Driver (IPW$$BS) sets the status bit NCBTPS3L. Next time, when the SD Subtask scans through the NCB-chain, the SD Subtask posts the I/O request complete and the READ CCW buffer is updated to contain an empty buffer. Thus the PNET Driver gets a chance to issue a new I/O request to send some data to the remote node. This happens for example if the PNET line driver sends
   a. a RIF record to start sending data to the remote node
   b. a PGR record to acknowledge the RIF record of the remote node (or NPGR record as negative acknowledgement)
   c. an EOT record to acknowledge the reception of a whole queue entry
   d. messages or commands via the console transmitter
   e. a SIGNOFF record
3. The FCS bytes of the local node have been changed by the PNET Driver and have to be sent to the remote node (see “Processing FCS Bytes” on page 233).

BCB Processing: The block control byte (BCB) is contained within the starting BSC frame for any record, even an empty block. The BCB contains a counter with values from 0 to 15 which starts again with 0 after 15. This counter is used to detect and correct sequence errors. There exists a BCB for the input buffers and another BCB for the output buffers.

Allthough the TCP/IP protocol has probably its own sequence checking, VSE/POWER maintains BCB's within the buffers sent and received via the TCP/IP connection. As usually no empty buffers are sent via the TCP/IP connection, the BCB within the TCP/IP buffers is different from the BCB contained in the current CTC buffers. Therefore VSE/POWER maintains BCB's for the CTC buffer (maintained by the PNET driver) and additional BCB's for the TCP/IP buffers (maintained by the SD Subtask).

The BCB for the TCP/IP send buffer is updated and sent to the remote node without any further processing.

The BCB for the TCP/IP receive buffer is checked for correctness and if invalid, an incorrect BCB is moved into the CTC buffer which causes the PNET driver:

1. to send a "BCB sequence error" record
2. to stop the node
3. to restart the node, if not suppressed by the NR option of the PSTART command.

The SD Subtask does not check the BCB within the CTC send buffer. Since the CTC buffer is used as the TCP/IP send buffer, the BCB within the CTC buffer is temporarily updated with the BCB for TCP/IP. When the CTC I/O is posted complete, the BCB of the CTC send buffer is reset to its original value.

The SD Subtask updates the BCB within the CTC read buffer with the expected value (NCBEBCB) updated by the PNET driver. Only if the SD Subtask detected a BCB error within the BCB of the TCP/IP receive buffer, the BCB of the CTC read buffer is updated with a value which causes a BCB error (processed by the PNET driver, see above).

If a BCB within the TCP/IP receive buffer indicates "reset BCB", the the BCB for TCP/IP send buffer is updated to the recived value, the BCB's of the CTC buffers remain undisturbed.

Control Flows:

Flow of I/O Processing: Referring to Figure 64 on page 207 which shows the modules and components involved in processing an I/O request and its corresponding socket calls after a PSTART command has been entered for the PNET TCP TD-subtask and its support module IPW$$TS, the SD-subtask and its support module IPW$$SS act in a similar manner.

Flow of Processing when Starting First Node: After the SD Subtask has been attached at PNET intialization on both nodes A and B, see Figure 71, the subtasks on both nodes prepare themselves to ACCEPT a CONNECT request from the other side. The CONNECT request is triggered by the PSTART PNET,NODEA command entered on NODE B. But the SD Subtask on NODE A rejects the connection, because the corresponding PSTART PNET,NODEB command has not yet been entered at NODE A. The SA-SENSE at node B is not yet posted complete.

```
                 NODE A                                                            NODE B
PLOAD                         SD Subtask                       SD Subtask                                PLOAD
PNET cmd                      SD-Driver/                       SD-Driver/                                PNET cmd
  *                           SS-Socket                        SS-Socket                                         -----
  |                                                                                                                |
  +----ATTACH-Subtask--->  INITAPI                                                                         *     * * *
                           GSKINIT                                                                         |     *   *
                           S0:=SOCKET                                                                      |     * T *
                           BIND(S0) to local port A                                                        |     * C *
                              +                                                                            |     * P *
                           LISTEN(S0)                        INITAPI         <-----ATTACH-Subtask--------+     * * *
                              +                              GSKINIT                                               |
                           SELECT read array + S0            B0:=SOCKET                                            |
                              +                              BIND(B0) to local port B                            * * *
                              +                                +                                                 * L *
                              +                              LISTEN(B0)                                          * I *
                              +                                +                                                 * S *
                              +                              SELECT read array +B0                               * T *
                              +                                +                                                 * E *
                              +                                +                               PNET              * N *
                              +                                +                             LD-Driver           * * *
                              +                                +                                       enter cmd --|--
                              +                                +                               + PSTART PNET,NODEA |
                              +                                +                               |                   |
                              +                                + C1:=SOCKET    <---------SA-SENSE cmd-code       * * *
                              +                                + IOCTL(C1)                                       *   *
                              +                                + CONNECT(C1) to A                                * T *
                              * :<-----post-read-array-for-S0-------                           .                 * C *
                              S1:=ACCEPT(S0)                   + IOCTL T(C1)                                     * P *
                                :---post-read-array-for-C1-->  + SELECT read array + C1        .                 * * *
                           SELECT read array + S1              +                                                   |
                                                               + SELECT write array + C1       .                   |
                                : <--post-read-array-for S1--  + SEND(C1) OPEN-CTL-rec         .                   |
                           RECV (S1)                           +  :                            .                   |
                                verify OPEN-CTL versus TCP-    +  :                            .                 * * *
                                PNODE entries of local NDT     +  :                            .                 * I *
                                and started TCP-nodes (NCB)    +                               .                 * N *
          >>>> assume 'no PSTART yet'                          + SELECT read array C1          .                 * I *
                           CLOSE(S1)                           +  :                            .                 * T *
                                  | --post-read-array-for-C1-> +  :                            .                 *   *
                                          with data-len=0----> + RECV(C1) returns:             .                 * * *
                              +                                + RC=-1,ERRNO=1124 or recv-len=0                    |
                           SELECT read array S0 (- S1)         +  :                            . => 1RC6I CONNECT  |
1RTHI NODE B AWAITING  <===== +                                +                                          PENDING  |
      CONNECTION              +                                +                                                   |
```

Figure 71. SD Subtask - Startup and First PSTART Command

Flow of Processing when Starting Second Node: With the first connection attempt rejected in Figure 71, and NODE B in 'CONNECTION PENDING FOR NODE A' state, the scenario continues with Figure 72 on page 237, when finally the missing PSTART PNET,NODEB command is entered on NODE A side.

```
                  NODE A                                                            NODE B

        PNET                   SD Subtask                       SD Subtask                    PNET
      LD-Driver                SD-Driver/                       SD-Driver/                  LD-Driver
                               SS-Socket                        SS-Socket

           (for entry, see precedeing figure 'Startup SD Subtask - Incoming CONNECT Request')
                              +                                 +                                                   |
           (SELECT read array incl ACCEPT sok)       (SELECT read array incl ACCEPT sok)                           |
                              +                                 +                                .               * * *
 enter cmd                    +                                 +                                .               * T *
 PSTART PNET,NODEB            +                                 +                                .               * C *
       |                      +                                 +                                .               * P *
    SA-SENSE cmd-code-------> + C2:=SOCKET                      +                                .               * * *
         .                    + IOCTL(C2)                       +                                .                  |
         .                    + CONNECT(C2)                     +                                .                  |
         .                    +      --post-read-array-for-B0-> +                                .                  |
         .                    + IOCTL(C2)                       +                                .                  |
         .                    + SELECT read array + C2          +                                .                  |
         .                    +    <--post-read-array-for-C2    B1:=ACCEPT(B0)                   .               * * *
         .                    + SELECT write array + C2         SELECT read array + B1           .               *   *
         .                    + SEND(C2) OPEN-CTL-rec ---->        :                             .               * I *
         .                    +  : --post-read-array-for B1-->     :                             .               * N *
         .                    +                                 RECV(B1)                         .               * I *
         .                    + SELECT read array incl C2       SELECT write array + B1          .               * T *
         .                    +  :       <-ACK-CTL-rec------      SEND(B1)                       .               * * *
         .                    +  : <--post-read-array-for-C2       :                             .                  |
         .                    + RECV                            +  :                             .                  |
         |<---'07'----for CTL-+  :                              SELECT read array incl B1        .                  |
         .                    +  :                              +  : ----for-no-cmd--'00'--> CONTROL                |
         .                    +  :                              +  :                         READ                   |
         .                    + GSKGETCIPHINF                   GSKGETCIPHINF                    .                  |
         .                    + GSKGETDNBYLAB                   GSKGETDNBYLAB                    .                  |
         .                    + GSKSSOCINIT                     GSKSSOCINIT                      .                  |
  >>>>>  . >>>>>>>>>>> all subsequent DATA exchange with TTB|TTR|DATA|TTR-EOB format only  <<<< . <<<<<<<<<<<<<     |
         :                    +  :                              +  :                             .                  |
  >>>>>  . >>>>>>>>>>> from now on the SELECT socket calls are not displayed any more      <<<< . <<<<<<<<<<<<<     |
         .                    +  :                              +  :                             .                  |
       *****                  +  :                              +  :                             .              -----
       SENSE                  +  :                              +                                                   |
       WRITE SOH ENQ -------->-+--:                             +         .                      |
         .                    + GSKSSOCWRITE -SOH-ENQ-buffer--> +  GSKSSOCREAD                   .                  |
         .                    +  :                              +  :   --------SOH ENQ ----- >  |                * * *
         .                    +  :                              +  :                          *****              * C *
         .                    +  :                              +  :                          SENSE              * T *
         .                    +  :                              +  :   <-------DLE ACK0------ WRITE              * C *
                              +  :       <---DLE-ACK0-from---   +  GSKSSOCWRITE                  .               * * *
       CONTROL                  GSKSSOCREAD                     +  :                             .                  |
       READ <------DLE ACK0---+- :                              +  :                             .                  |
       *****                  +  :                              +  :                             .               * * *
         .                    +  :                              +  :                             .               *   *
  >>> from now on the DATA in TTB|TTR|DATA|.... has NJE-CTC buffer format: DLE|STX|BCB|FCS|FCS|RCB....    <<<<<< * I *
         .                    +  :                              +  :                             .               * N *
       SENSE                  +  :                              +  :                          CONTROL            * I *
       WRITE 'I' signon rec->-+--:                              +  :                             .               * T *
         .                    +  GSKSSOCWRITE 'I'-signon-rec--> +  GSKSSOCREAD                   .               * * *
         .                    +  :                              +  : --------'I'-signon rec->-READ                  |
         .                    +  :                              +  :                          ***=> MSG 1RB3I SIGNED ON
         .                    +  :                              +  :                          SENSE                 |
         .                    +                                 +  :  <------'J' signon resp- WRITE                 |
                              +  :      <-'J'-signon response-  +-GSKSSOCWRITE                   .                  |
                              +  GSKSSOCREAD                                                     .                  |
       CONTROL                +                                                                  .                  |
       READ <-'J'-signon resp-+--:                              +  :                             .                  |
1RB3I<=*****                  +  :                              +  :                             .                  |
SIGNED ON                     +  :                              +  :                             .                  |
   ---------------- N O D E S -------- A R E -------- S I G N E D ---------- O N -----------------------------------
```

Figure 72. SD Subtask - Second PSTART Command and Signon Complete

Flow of Processing the Idling State

The idling state is entered, if both nodes issued a CTC I/O request to send an empty buffer. Since empty buffers are not sent via the TCP/IP connection, both CTC I/O requests are not posted complete, and the nodes are set into "idling state".

If node A wants to send a record (containing a message for example) to node B, the record is queued as first buffer in the to-be-sent-queue and the status in the NCB is updated to leave the idling state. Next time, the SD Subtask is posted due to its own timer interval, the SD Subtask processes the new status of the NCB. Since no data has been received via the TCP/IP communication, the SD Subtask completes the CTC I/O and passes back an empty buffer. Thereafter the PNET driver starts a CTC I/O request to send the record to node B. The SD Subtask processes the CTC I/O request and issues a socket call GSKSSOCWRITE to send the record via the TCP/IP communication. The SD Subtask completes the CTC I/O request immediately and passes back an empty buffer, in order to let the PNET driver send some more data.

```
                  NODE A                                                            NODE B

        PNET                   SD Subtask                       SD Subtask                     PNET
      LD-Driver                SD-Driver/                       SD-Driver/                   LD-Driver
                               SS-Socket                        SS-Socket

                              +  :                              +  :
  >>>>>  . >>>>>>>>>>> the SELECT socket calls are not displayed in this figure       <<<< . <<<<<<<<<<<<<
                              +  :                              +  :                          *****
                              +  :                              +  :                          SENSE
                              +  :                              +  : <--- empty buffer -----  WRITE
                              +  :                              +  no GSKSSOCWRITE of empty bfr .
                              +  :                              +  :                            .
                              +  :                              +  :                          wait for I/O completion
       *****                  +  :                              +  :                            .
       SENSE                  +  :                              +  :                            .
       WRITE -- empty buffer --->no GSKSSOCWRITE of empty bfr | +  :                            .
         .                    +                                 +  :                            .
wait for I/O completion       +  :                              +  :                            .
         .                    +  :                              +  :                            .
         .                    +  :                              +  :                            .
         .             --------- Idling state: both CTC I/O not yet complete -------            .
         .                    +  :                              +  :                            .
         .                    +  :                              +  :                            .
'message to be sent'==|       +  :                              +  :                            .
         .            |========> update NCB status              +  :                            .
       CONTROL                +  :                              +  :                            .
       READ <--'empty buffer'----complete I/O                   +  :                            .
       *****                  +  :                              +  :                            .
       SENSE                  +  :                              +  :                            .
       WRITE --- message ------->:                              +  :                            .
         .                    +  GSKSSOCWRITE message ----|     +  :                            .
         .                    +  :--post-read-array-------|     +  :                            .
       CONTROL                +  :                        |     +  :                            .
       READ <-- empty buffer ----complete I/O (may be ..  |     +  :                            .
       *****                  +  : ..more data to send)   |     +  :                            .
       SENSE                  +  :                        |     +  :                            .
       WRITE -- empty buffer --->no GSKSSOCWRITE of empty | bfr +  :                            .
         .                    +  :                        |     +  :                            .
wait for I/O completion       +  :                        |     +  :                            .
         .                    +  :                        |---> +  GSKSSOCREAD                  .
         .                    +  :                              +  :                          CONTROL
         .                    +  :                              +  : --- message -----------> READ
         .                    +  :                              +  :                          *****
         .                    +  :                              +  :                          SENSE
         .                    +  :                              +  : <--- empty buffer -----  WRITE
         .                    +  :                              +  no GSKSSOCWRITE of empty bfr .
         .                    +  :                              +  :                            .
         .                    +  :                              +  :                          wait for I/O completion
         .                    +  :                              +  :                            .
         .             --------- Idling state: both CTC I/O not yet complete -------            .
```

Figure 73. SD Subtask - Processing the Idling State

TCP/IP Driver Subtask Mainline (IPW$$SD):

Overview: The module IPW$$SD is a copy of module IPW$$TD, which has been adapted to process SSL nodes. Labels of common routines have not been changed, which means IPW$$SD contains a lot of labels starting with the characters TD. Similarly, IPW$$SD addresses a SD Subtask specific control block, the SDCB which is a copy of the control block used by the TD Subtask. Although the control block is called SDCB, the fields of the SDCB start with the characters TDCB. The module IPW$$SD consists of following parts:

SDCS Initialization after being attached

TDMAIN Loops forever untill a stopcode is set. Following processing occurs within this loop:

TDTERM	Terminates TCP/IP interface and SSL feature if necessary
TDINITAP	Initializes TCP/IP interface and SSL feature if necessary
TDPASSOK	Processes as server in "passive mode"
TDNBLPIN	Loops through the NCB-chain to process all TCP-nodes (in 'active mode')
TDPOSLDR	Posts PNET Driver if necessary
TDDETACH	Detach itself (the SD Subtask) if necessary
TDTIMSET	Issues timer interval using TQE within subtask control block (SDCB)
TDWAITDS	Issues VSE/ESA wait macro If posted, starts loop from the beginning

TDSBATDY Tidy-up routine in case of abnormal termination, called by IPW$$AT.

SDCS - Initialization

The part SDCS performs following initialization steps after being attached:

1. Initialize base register
2. Initialize register for save area
3. Initialize register for save area used by macro IPW$IDM
4. Drop protection key zero in order to run in parallel mode, if multiprocessor support is available
5. Initialize areas for the timer services within SDCB
6. Initialize TCP/IP workareas within SDCB
    a. Clear workarea within module IPW$$SD
    b. Clear ITP workarea for passive mode
    c. Clear local IP-address which is updated later on by the socket call GETHOSTID
    d. Clear SSL workareas
    e. Set trace option on for socket calls issued during initialization
    f. Set request 'initialize interface to TCP/IP'
7. Update task identifiers (name and VSE/ESA subtask identifier) in SDCB
8. Anchor address of tidy-up routine TDSBATDY in SDCB, which is used for abnormal termination in IPW$$AT
9. Initialize timer services (use macro IPW$TTS with operand STXIT=YES)
10. Initialize maximum socket number which can be used for SSL nodes

TDMAIN - Mainloop

Overview: The part TDMAIN contains the mainloop of the SD Subtask, which is left only if some stop conditions cause the detach of the SD Subtask. In case of an abnormal termination this loop is not entered, but the tidy-up routine TDSBATDY is entered directly from IPW$$AT. At the end of the mainloop, the VSE/ESA WAIT macro is issued using as ECB the field TDCBECB within the SDCB. TDCBECB is posted due to the events described above (see “Posting Events” on page 227). When the SD Subtask is posted, the mainloop is started again at label TDMAIN. At the beginning of the mainloop following steps are performed:

1. Clear the ECB (TDCBECB)
2. Call timer services to process expired timer intervals (use macro IPW$TTS with operand PROCESS=YES)

Following steps are part of the mainloop:

TDTERM: The SSL feature is terminated by issuing the socket call GSKUNINIT, if once the socket call GSKINIT completed successfully. The interface to TCP/IP is terminated by issuing the socket call TERMAPI, if once the socket call INITAPI completed successfully. It is terminated according to the stop condition: if the interface has to be terminated immediately, the socket call TERMAPI is issued at once. In all other cases, the TERMAPI is postponed untill all SSL nodes (TDCBNONU must contain 0) have been stopped and the passive connection has been stopped. In case the initialization (socket call INITAPI) has failed, no socket call TERMAPI is issued. In any case, any outstanding message of type A (anchored in TDCBMSGD) is deleted via macro IPW$GTS using the operand DOM. Thereafter a terminating message, either 1RT8I or 1RTSI, is issued via macro IPW$GTS with the operand MSG.

TDINITAP: The interface to TCP/IP is initialized by issuing the socket call INITAPI, as well as the SSL feature by issuing the socket call GSKINIT, but only if no stop condition is set. In addition the status "issue startup-message" (TDCBS1SM) is set. The startup-message 1RT7I is issued once for each initial socket call (SOCKET, GETHOSTID, BIND, LISTEN). Thereafter the initialzation process for the passive mode is entered by entering the routine TDINIGSC directly.

TDPASSOK: The part TDPASSOK contains the code for the SD Subtask to run as a server in passive mode. Following steps are performed:

1. If the interface to TCP/IP is not available or the passive mode has been stopped, passive mode processing is bypassed and processing continues with searching through the NCB-chain.
2. If the passive mode has to be stopped, processing continues with closing the passive connection (TDPASCLS)
3. If the passive mode has been started (a CONNECT request from a remote node has been received), but the initial contact (the exchange of the OPEN and ACK or NAK control records) did not complete in time, the processing continues with closing the passive connection (TDPASTIO)
4. If a special event (TDNTPS4P) has to be processed, processing continues with the routine for this event (TDNTPWPO). At this point only one event may happen:
   a. When receiving a CONNECT, the NCB chain had to be scanned for matching definitions. For this purpose the PNCB had to be locked. If the locking of the PNCB had failed, the status (TDNTPS4P) is updated to reenter the routine (TDPASVLF) to lock the PNCB.
5. In all other cases processing continues with the routine to issue a socket call (TDSBSOCK).

When routine TDSBSOCK is entered, register 1 points to the parameter area (TDNTPDS) which contains all necessary information to issue a socket call, namely:

1. the type of socket call (TDNTPSC)
2. the return addresses (TDNTPS00, TDNTPS04, TDNTPS08, TDNTPS0C) for the routine TDSBSOCK according to the returncode (0,4,8,12) set by the service routine (IPW$$SS)

The parameters are set by the routines initializing the socket call before entering the routine TDPASSOK. This means the routine TDPASSOK is entered directly by subroutines to issuing a socket call and thereafter, once the socket call has been issued, by the mainline to check for the completion of the socket call.

Following socket calls are issued in passive mode

1. The first sequence of socket calls is issued to prepare the passive mode.

   The return address (TDNTPS00) for the returncode 0 is set in each routine before the socket call is issued. The return addresses (TDNTPS04, TDNTPS08, TDNTPS0C) for the error conditions are set just once and have been initialized in routine TDINITAP before issueing the socket call INITAPI. TDINITAP passes control to routine TDINIGSC to start the initial sequence of socket calls.

   Following routines are involved in preparing passive mode processing:

   TDINIGSC Issue the socket call SOCKET to allocate the necessary control blocks for the passive mode. The parameter area is initialized with the port number (TDNSCBPT) out of the NDT and the family type (TDNSCBFM) is set to 2. Returned is the socket descriptor

(TDNSCRC), a number which is saved into TDNSCSOD, because it must be referenced by all following socket calls. In addition the status "TCP/IP interface available" (TDCBS1IA) is set, because the socket call SOCKET was the first, which has been passed through all the TCP/IP layers. If this socket calls fails, the type A message 1RTJA is issued, which will be deleted if the interface has been established successfully or the SD Subtask is stopped. If the returncode makes a retry meaningful, the socket call is retried every 20 seconds.

TDINIGIP Issue the socket call GETHOSTID to get the binary IP-address of the local node and update TDCBSCIR with the readable IP-address.

TDINIBID Issue the socket call BIND to link the socket with a port number.

TDINILIS Issue the socket call LISTEN to prepare the socket for the next socket call ACCEPT. The status "issue startup-message" (TDCBS1SM) is reset and any outstanding message of type A (anchored in TDCBMSGD) is deleted via macro IPW$GTS using the operand DOM.

2. The next sequence of socket calls is issued to process incoming CONNECT request from remote nodes and complete the initial contact (see “Processing as Server and Client” on page 228) The return addresses (TDNTPS00, TDNTPS04, TDNTPS08, TDNTPS0C) are set in the routines to the appropriate values.

   Following routines are involved in passive mode processing:

   TDPASACC Issue the socket call ACCEPT to get posted whenever a CONNECT request is received. The posted ACCEPT returns a new socket descriptor (TDNSCSOD), because new control blocks have been allocated by the TCP/IP layers when the CONNECT has been received. This new socket descriptor is now used for all following socket calls. The socket descriptor used for the socket call ACCEPT has been saved into TDSVSCSD in routine TDINIGSC and is used for all new ACCEPT socket calls. The received binary IP-address of the remote node is translated into readable format and put into TDNTPIPC.

   In addition a timer interval is set for 5 minutes. Within this time limit the initial contact must be completed, otherwise the local node stops the connection by issuing a socket call CLOSE and message 1RTGI. This avoids that a hanging connection prevents that other connections can not be established, because one CONNECT request after the other is processed sequentially.

   TDPASGHN Issue the socket call GETHOSTBYADDR to get a logical hostname. The previously posted ACCEPT returned just a binary IP-address, but no node name. When checking the network definition table (NDT) both IP-addresses, the binary IP-address and the logical hostname, are used to check whether a node has been defined for one of these addresses.

   If no matching node has been found in the NDT, the message 1RT3I is issued. If a logical hostname was found for the received binary IP-address, the message 1RTBI is issued in addition.

   TDPASRCV Issue the socket call RECV to receive an OPEN control record.

   Following error messages are issued:

   1RT4I The control record was not of type OPEN. The connection is stopped by issuing a socket call CLOSE. No NAK control record is sent.

   1RT5I The control record contained invalid information, which might be the FROM node-id or FROM IP-address or the TO node-id or TO IP-address. The connection is stopped by issuing a socket call CLOSE after a NAK control record with RC=1 has been sent.

1RTEI The binary IP-address was not used for the node-id found in the NDT according to the binary IP-address received by the CONNECT request. The new connection is stopped by sending a NAK control record with RC=1 and issuing a socket call CLOSE.

1RTBI The logical hostname found due to the binary IP-address of the CONNECT request was not used for the node-id found in the NDT. The new connection is stopped by sending a NAK control record with RC=1 and issuing a socket call CLOSE.

1RTHI A PSTART command has not yet been issued for the node-id. The new connection is stopped by issuing a socket call CLOSE without sending a NAK control record.

1RTVI A PSTART command has already been issued for the node-id. One of the following situation has occurred:

a. The SD Subtask is just starting a connection for this node-id in active mode. Therefore the new connection, started by the passive mode, is stopped by sending a NAK control record with RC=3 and issuing a socket call CLOSE. The status of the connection starting in active mode is not changed at all and continues its normal flow.
b. The TCP/IP connection for the node-id has been established some time ago successfully. Therefore the new connection, started by the passive mode, is stopped by sending a NAK control record with RC=2 and issuing a socket call CLOSE. The already active connection is stopped as well by setting the stopcode to line-error.

1RTFI The received control record is displayed in hexadecimal format. This message is displayed in addition to one of the previous messages (1RT4I or 1RT5I).

1RV3I The remote node does not use the SSL feature (a control record with type "OPEN " instead of "OPEN SSL" has been received). The connection is stopped by issuing a socket call CLOSE after a NAK control record with type "NAK SSL" and RC=1 has been sent.

TDPASSND Issue the socket call SEND to send the ACK or NAK control record. If an ACK control record has been sent, the status bytes within the NCB are updated to continue processing for this node (in active mode) using the TCP/IP connection established in passive mode. If a NAK control record has been sent, the passive connection is closed by issuing a socket call CLOSE. In both cases the workarea for the passive connection within the SDCB is cleared to process new incoming CONNECT requests. Processing continues in routine TDPASACC with issuing a socket call ACCEPT.

TDNBLPIN: The part TDNBLPIN contains the code for the SD Subtask to run as a client in active mode. Following steps are performed (for more details see “TDNBLPIN - Loop through NCB-Chain (Details)” on page 244):

TDNBLPIN Init search through NCB chain

TDNBLPNX For each SSL node in NCB chain:

TDNBCLOS Close connection if necessary

TDNBINIT Start initial contact if necessary

TDNBSSLI Initialize SSL feature for this connection

TDNBWAIT Process next NCB if processing for current NCB is postponed

TDNBIOST Start processing of CTC I/O request: translate CCW's

TDNBIOSN Process SENSE CCW

TDNBIOWR Process WRITE CCW

TDNBIOCL Process CONTROL CCW

TDNBIORD Process READ CCW

TDNBFCSW Init wait timer interval if FCS signals "hold stream(s)"

TDNBIDLG Leave idling state if necessary

TDNBIOZ0 Complete I/O if necessary

TDNBTIME Init timer interval for this NCB if necessary

TDPOSLDR: If the status "post PNET Driver" (TDCBA3PL) is set, the ECB (TCEB) of the PNET Driver is updated and the main-ECB (PAEB within the CAT) of the VSE/POWER maintask is posted by using the VSE/ESA macro POST. TDCBA3PL has been set, if an I/O request for a node has been completed and an input buffer has been queued to the PNET Driver TCB in routine TDNBIOZ0.

TDDETACH: If the status "detach task" (TDCBA1DT) has been set in routine TDTERM after the socket call TERMAPI has been issued, the SD Subtask is now detached by issuing the VSE/ESA macro DETACH. The SD Subtask can not be detached immediately in routine TDTERM, because in case of immediate termination of the SD Subtask, the status bytes for the SSL nodes have to be set first by entering routine TDNBLPIN. Before detaching the SD Subtask, the timer service is called by using the macro IPW$TTS with the operand CANCEL to cancel an outstanding timer interval using the TQE within the SDCB.

TDTIMSET: First the timer service is called by using the macro IPW$TTS with the operand CANCEL to cancel an outstanding timer interval using the TQE within the SDCB. Thereafter the timer service is called by using the macro IPW$TTS with the operand TIME to issue a new timer interval using the TQE within the SDCB.

TDWAITDS: If the status "omit WAIT" (TDCBS2NW) is not set, the VSE/ESA macro WAIT is issued using the ECB (TDCBECB) within the SDCB. TDCBS1NW is set in case the TCP/IP interface is no longer available in error routines TDINITII and TDPASSTI. In these cases a WAIT is not necessary, the mainloop is directly reentered at label TDMAIN to terminate and detach the SD Subtask.

TDNBLPIN - Loop through NCB-Chain (Details)

Following steps are processed for each NCB of the NCB chain.

TDNBLPIN: Before searching through the NCB chain for a SSL node, the PNCB is locked using the test and set (TS) instruction. If the locking is successful, the lockword is updated with "SSL" to identify the SD Subtask as owner of the PNCB. If the locking is unsuccessful, the SD Subtask waits for 1 second using the macro IPW$TTS with the operand "REACTIVATE" before retrying to lock the PNCB.

The PNCB is unlocked, if the end of the PNCB chain is reached or if a SSL node is found for which the TCP/IP connection has not yet been closed (NCBTPS1F).

TDNBCLOS: Call routine TDSBSTST to terminate the SSL feature, which issues the socket calls, if necessary:

GSKFREEMEM
to free memory that was passed to the application on a previous call to an SSL function.

GSKSSOCCLOSE
to close secure socket connection and free all SSL for VSE/ESA resources for that connection.

Depending on the status bytes within the NCB (NCBTPSTx):

1. A message is issued explaining the reason why the the SSL connection has been closed
2. If a SIGNOFF record has been received, processing continues at TDNBTIME to terminate the TCP/IP connection
3. If TCP/IP connection to be restarted:
   a. Clear the TCP/IP workarea (starting at NCBTPDS)
   b. Processing continues at TDNBTIME to init timer interval before restarting
4. In all other cases, processing continues to complete I/O (TDNBIOZ0)

TDNBINIT: The routine TDNBINGS (for more details see “TDNBINGS - Establish Initial Contact (Details)” on page 247) to establish the initial contact is entered:

1. If the initial contact is not yet complete
2. And if no wait is issued
3. And if a wait has expired in case a wait has been iussed earlier
4. And if a connect request from the remote node is not processed by the passive mode
5. And if the TCP/IP interface is available

TDNBSSLI: Call routine TDNBSSL0 to initialize the SSL feature, which issues the socket calls:

GSKGETCIPHINF
to request cipher related information for SSL for VSE/ESA.

GSKGETDNBYLAB
to get the label for a key in a key database file. application on a previous call to an SSL function.

GSKSSOCINIT
to initialize the data areas necessary for SSL for VSE/ESA to initiate or accept a secure socket connection.

TDNBWAIT: If a timer interval has been set and is not yet expired, processing continues with the next NCB. If a timer interval has been set and has expired:
if received or transmitted FCS bytes of CTC buffers still hold a stream, status "complete I/O" is set to give the PNET driver a chance to update FCS bytes (see “TDNBFCSW” on page 247).

TDNBIOST: If status is "start I/O processing", initialize address of last CCW processed and issue trace message containing CCW-chain, if console trace is started.

TDNBIOSN - Loop through CCW-Chain

Start processing of CCW-chain, if TCP/IP connection not closed:

1. If stand-alone SENSE CCW, continue to finish CCW processing (TDNBIOLC).
2. Otherwise update sense byte with op-code CONTROL and continue processing with next CCW (TDNBIONC).

TDNBIOWR: If WRITE CCW:

1. If CTC I/O already once processed, continue processing next CCW (meaningful if CTC buffer contained empty buffer).
2. If signon complete, FCS bytes did not change and CTC buffer contains empty buffer, omit socket call GSKSOCWRITE and continue processing next CCW (TDNBIONC).
3. Update BCB in TCP/IP send buffer (as the CTC write buffer is part of the TCP/IP send buffer, the BCB of the CTC buffer is first saved)
4. FCS bytes of CTC write buffer remain unchanged in TCP/IP send buffer
5. Update TTB, starting TTR and TTR-EOB in TCP/IP send buffer

6. Update address and length of data and return addresses according to the return code of the socket call
7. Call subroutine to issue send request
8. If socket call GSKSSOCWRITE completed:
   a. If not all bytes sent, update address and length of data and continue processing with subroutine call to issue send request
   b. Restore BCB in CTC write buffer
   c. If SOH-ENQ or ACK sent, do not update status "complete CTC I/O", but wait till SIGNON record has been received
   d. If SIGNOFF record sent, continue processing with closing the TCP/IP connection (TDNBCLOS)
   e. In all other cases update status "complete CTC I/O" and continue processing next CCW (TDNBIONC)

TDNBIOCL: If CONTROL CCW, continue processing next CCW (TDNBIONC).

TDNBIORD: If READ CCW:

1. If CTC I/O not yet processed once and FCS bytes hold at least one strean and console trace is started, issue trace mesage with FCS bytes of CTC write buffer.
2. If FCS bytes hold all streams, omit socket call GSKSSOCREAD (if FCS bytes hold just one stream, we trust remote node and assume the received buffer will never be for the suspended stream. RSCS processes the FCS bytes too late and sends buffer even for a suspended stream, but the PNET driver ignores this protocol violation and continues processing without any error indication)
3. If no data left over from last socket call GSKSSOCREAD, continue with starting socket call GSKSSOCREAD
4. If received data in TCP/IP receive buffer contains all bytes for a CTC read buffer (without TTB and TTR):
   a. Move data from TCP/IP receive buffer to CTC read buffer
   b. If BCB of TCP/IP receive buffer is invalid, update BCB of CTC read buffer to an invalid value which forces a BCB sequence error issued by the PNET driver (node is stopped and restarted)
   c. If console trace is started, issue message displaying TTR and first 12 bytes of CTC read buffer
   d. If socket call GSKSSOCWRITE started and not yet complete, omit setting of status "complete CTC I/O"
   e. Continue processing next CCW (TDNBIONC).
5. Start processing of socket call GSKSSOCREAD
   a. If any bytes left after having moved some data from TCP/IP receive buffer into CTC read buffer, move left data to begin of TCP/IP receive buffer and display data via trace message, if console trace started
   b. Update address and length of data and return addresses according to the return code of the socket call
   c. Call subroutine to issue receive request
   d. If socket call GSKSSOCREAD completed:
      1) If no bytes received, continue processing with closing the TCP/IP connection
      2) Continue processing above with checking, if all bytes for a CTC read buffer (without TTB and TTR) have been received
   e. If socket call GSKSSOCREAD should be retried (return code = 4 of socket call):
      1) If signon not yet complete, continue processing with next CCW (and wait till all data received)
      2) If CTC I/O to be completed (send socket call completed), continue processing with next CCW (and do not wait till receive complete)
      3) If no data received and if no send socket call outstanding, set status "idling"
      4) Continue processing next CCW (TDNBIONC).

TDNBIONC - processing next CCW: If CCW is not last CCW, update address of last CCW processed (NCBLCCW) and continue to process the CCW (TDNBIOSN)

If CCW is last CCW, update status "I/O once processed".

TDNBFCSW: If signon is complete (FCS bytes are contained within CTC buffers) and received or transmitted FCS bytes hold at least one stream and the CTC I/O can not be completed, continue processing to initiate wait for a timer interval of 20 seconds to suspend processing for this node (see "TDNBWAIT" on page 245).

TDNBIDLG: If status is "leave idling state" and "idling", set status "complete I/O".

TDNBIOZ0: To complete the CTC I/O, following steps are performed:

1. If status is not "line busy" and not "close line", issue IDUMP macro and message 1RTLI.
2. Update address of last processed CCW (NCBLCCW) to point after last processed CCW
3. Update CCB status with "channel and device end"
4. If no data received via TCP/IP connection, build empty buffer
5. Update residual count in CCB
6. If console trace started, issue trace message to display CCB, address of last processed CCW, and some status bytes
7. Queue buffer to channel end queue of PNET driver via compare and swap instruction
8. If SIGNOFF record received, continue processing with closing the TCP/IP connection (TDNBCLOS)
9. If TCP/IP connection closed and to be restarted, init timer interval for restart
10. If TCP/IP connection closed and not to be restarted, set status "terminate connection"

TDNBTIME: If status is "terminate connection":

1. Cancel any outstanding TQE for this NCB and update status "TCP/IP connection finished" (NCBTPS1F, used by PNET driver to start final node clean up).
2. If no SIGNOFF record received nor sent, update status with TCP/IP error.
3. Continue processing next NCB (TDNBNEXT).

If timer interval to set, initiate timer interval using value (in NCBTPTIV) previously set and update status "waiting for expiration of timer interval" (NCBTPS4W).

TDNBINGS - Establish Initial Contact (Details)

The SD Subtask performs following steps to establish the initial contact as a client in active mode:

TDNBINGS Issue the socket call SOCKET to allocate the necessary control blocks for a connection in active mode.

TDNBINGH If a logical IP address has been specified for the NCB, issue the socket call GETHOSTBYNAME to get a binary IP address.

TDNBINIO Issue the socket call IOCTL to set nonblocking mode for connection.

TDNBINCO Issue the socket call CONNECT to send a connect request

TDNBINIO Issue the socket call IOCTL to set blocking mode again for connection.

TDNBINSR If CONNECT completed successfully, issue the socket call GSKSSOCWRITE to send an OPEN control record to the remote node.

TDNBINRR If GSKSSOCWRITE completed successfully, issue the socket call GSKSSOCREAD to receive an ACK control record from the remote node.

If no ACK nor NAK control record has been received, issue messages 1RTDI and 1RTFI, and continue processing to close the TCP/IP connection.

If the ACK or NAK control record contains incorrect values, issue messages 1RT5I and 1RTFI, and continue processing to close the TCP/IP connection.

If a NAK control record has been received, issue message 1RT6I and continue processing to close the TCP/IP connection. If a NAK control record with RC=3 has been received, update status "restart TCP/IP connection" (NCBTPS1R).

If an ACK control record has been received:

1. Set status "initial contact complete"
2. Set status "complete I/O"
3. Update sense bytes with CCW op-code CONTROL
4. Continue processing with process CTC I/O (TDNBIOST).

IPW$$SD - Tidy-up Routine - TDSBATDY: This routine is called by module IPW$$AT in case of an abnormal termination of the SD Subtask. Following steps are performed:

1. The timer service of the VSE/ESA supervisor is terminated by issuing the VSE/ESA macro STXIT with the operand IT. This terminates any timer service established by one of the TCP/IP layers which have been called when a socket call is issued.
2. The anchor point (TDCBTQEA) for the timer service is cleared.
3. The following resources are unlocked by clearing the lockword:

   TIBLCK The trace information block which might have been locked because trace entries have been written into the storage trace area

   CAAB The asynchronous service anchor block which might have been locked because the filled up storage trace area had to be written to disk
4. The main-ECB (PAEB within the CAT) of the VSE/POWER maintask is posted by using the VSE/ESA macro POST in order to resume processing of any task waiting for one of the above resources.
5. Any outstanding message of type A (anchored in TDCBMSGD) is deleted via macro IPW$GTS using the operand DOM.
6. In order to stop all SSL nodes the PNCB is locked. If the PNCB can not be locked, a wait for a second is issued by using the macro IPW$TTS with the operand REACTIVATE to be posted after 1 second
7. In order to stop all SSL nodes as fast as possible the following information is set
   a. NCBTPS4C - Socket call CLOSE has been issued
   b. NCBTPS1F - TCP/IP connection is finished
   c. NCBF1BY - No I/O request outstanding
   d. NCBTTCL - Line error
   e. NCBLNSR - Close line, used by the activity-process of the PNET Driver (IPW$$LD3)
   f. The post bit within the ECB (TCEB) of the PNET Driver is set.
8. The PNCB is unlocked and the main-ECB (PAEB within the CAT) of the VSE/POWER maintask is posted by using the VSE/ESA macro POST.
9. If the SSL feature was initialized, the socket call GSKUNINIT is issued.
10. If the TCP/IP interface was available, the socket call TERMAPI is issued.
11. Issue message 1RT8I TCP/IP: INTERFACE NOT AVAILABLE
12. Reset status "SSL feature initialized", "interface available" and "interface once available", other status information is reset when entering the initialization process (SDCS).
13. Reload saved registers and return to caller (IPW$$AT).

TCP/SSL Driver Subtask Services Interface Macros (IPW$$SS): In order to separate subtask service functions from the TCP/SSL driver subtask mainline, the following services are provided (see Appendix C, “VSE/POWER Internal Macros” on page 747 for more details):

1. EZASMI Socketcall Support
2. Message Support
3. Timer Interval Interrupt Supprt
4. EZASMI Socketcall Error Checking Support

In addition, the Service Support module has its own internal tracing.

Subtask EZASMI Socketcall Support (IPW$ITS Macro): Using the IPW$ITS macro, the subtask may invoke the EZASMI API and check for error conditions afterwards.

IPW$ITS PARMS=socketcall: Using the IPW$ITS macro, the subtask may invoke the EZASMI API for the following socketcalls:

- ACCEPT
- BIND
- CANCEL
- CLOSE
- CONNECT
- GETHOSTID
- GETHOSTBYADDR
- GETHOSTBYNAME
- GSKINIT
- GSKUNINIT
- GSKGETDNBYLAB
- GSKFREEMEM
- GSKSSOCINIT
- GSKSSOCREAD
- GSKSSOCWRITE
- GSKSSOCCLOSE
- GSKSSOCRESET
- GSKGETCIPHINF
- INITAPI
- IOCTL
- LISTEN
- RECEIVE
- SELECT(Read)
- SELECT(Write)
- SEND
- SOCKET
- TERMAPI

The EZASMI interface is invoked in 31-bit mode.

Internally the IPW$$SS module will invoke the IPW$ITS CKRC=YES macro to check the EZASMI socketcall for any immediate error return.

IPW$ITS CKRC=YES: This macro is called internally in the IPW$$SS module to check for immediately returned EZASMI API access errors, and also by the IPW$$SD module to check for errors following EZASMI API ECB posting.

Subtask Message Support (IPW$GTS Macro): Since a VSE/POWER subtask cannot use the messaging support available to the maintask, the following functions are provided with their own access macro.

IPW$GTS MSG=msgid: This access macro allows the caller to specify the message equate "msgid" of a message defined by the IPW$GMM macro in the IPW$$MM module. The message will be issued in the same way as for the maintask, using the WTO macro and providing message substitution and message squeezing via the IPW$$MX module.

IPW$GTS MSG=TRACE: This access macro allows the caller to issue a PNET Driver Subtask trace message (1RTTI).

IPW$GTS DOM=(R1): This access macro allows the caller to delete a console message issued previously by the IPW$GTS MSG= macro.

Subtask Timer Interval Interrupt Support (IPW$TTS Macro): The are various support access macros:

IPW$TTS STXIT=YES: This access macro initializes the VSE Timer STXIT interface for the SETIME macro used for the other support macros.

IPW$TTS TIME=(Rx),TQE=: This access macro allows the caller to indicate a timer interval in tenths of a second following which an ECB is posted in the indicated TQE element and the Driver Subtask is also posted.

IPW$TTS CANCEL=YES,TQE=: The caller indicates that a previous IPW$TTS TIME= request is to be cancelled.

IPW$TTS PROCESS=YES: This access macro is called by the Driver Subtask following posting.

IPW$TTS WAIT=(Rx): This access macro allows the Driver Subtask to indicate it wishes to go into a wait state until it is posted by either the expiration of a SETIME interval request for the WAIT= interval (in tenths of a second), or by any other event which may occur sooner, with the register Rx containing the interval value.

IPW$TTS WAIT=(Rx,REACTIVATE): This access macro allows the Driver Subtask to indicate it wishes to go into a wait state as for the IPW$TTS WAIT(Rx) macro, and additionally the macros IPW$TTS STXIT=YES and IPW$TTS PROCESS=YES are called immediately following.

Subtask Support Internal Trace: The support module IPW$$SS has its own internal tracing area. Each module entry and exit is recorded in the trace area with an eye catcher and register contents. At the end of the IPW$$SS module, beginning at the eye catcher "LAST ENTRY =" lies the trace area. The layout area is:

- eye catcher "LAST ACCESS="
- address of the last trace entry that was recorded (4 bytes)
- eye catcher "LAST BRANCH="
- address to which the module last exited
- (80 byte entries) with the layout:
  - eye catcher describing the entry (16 bytes)
  - contents of the registers 0 to 15
- eye catcher "$$SSBUF END"

PNET SNA Interface to VTAM

Opening the VTAM Interface: The interface is opened when the operator enters a PSTART PNET,nodeid command. If the VTAM interface has not yet been opened (first PSTART command), the PNET driver attaches the routine IPW$$S1 as a VSE/AF subtask.

PNET VTAM OPEN/CLOSE Subtask: The IPW$$S1 routine is invoked in order to identify VSE/POWER to VTAM as an application program. It is attached as an independent VSE subtask by the PNET driver and remains active until the network is shutdown.

The IPW$$S1 routine performs following functions:

- Identifies VSE/POWER as an application program to VTAM by issuing the VTAM OPEN request. The IPW$$S1 routine builds an VTAM ACB in the VDCB. An VTAM ACB model has been predefined in the routines CSECT. The address of the own APPLID (located in the PNCB) as well as the address of the VTAM Exit Routines definition (VTAM macro EXLST, defined in routine IPW$$SE) are stored into the VTAM ACB.
- Enables the necessary VTAM Exit Routines so that session requests can be received from remote nodes. This is done by issuing the VTAM SETLOGON START request.
- Disables the VTAM Exit Routines before shutting down the network by issuing the VTAM SETLOGON QUIESCE request.
- Dissociates VSE/POWER from VTAM by closing the VTAM interface (VTAM CLOSE).

If the VTAM interface could not be opened, error message 1RD2I (including the reason code) is issued. The reason code is taken from the VTAM ACB Error Field. The PNET driver ECB as well as the VSE/POWER Master ECB are posted and the subtask is detached from VSE/AF.

Enable VTAM SCIP Exit: VSE/POWER is known to VTAM as an application program after the VTAM interface has been opened successfully. VSE/POWER can now act as a primary application program, i.e. it can send session requests to other nodes. The SETLOGON OPTCD=START request is issued so that VSE/POWER can receive session requests from other nodes and thus can act as a secondary application program.

If an error occurs during the SETLOGON processing, then VSE/POWER cannot act as a secondary application program, i.e. no session requests issued by remote nodes can be received. Error message 1RD3I is issued including the reason codes causing the failure. The reason codes are taken from the VTAM RPL RTNCD and FDBK2 fields. The VTAM interface is closed, the PNET driver ECB is posted and the subtask is detached from VSE.

The VSE subtask IPW$$S1 posts the PNET driver ECB and issues a VSE/AF WAIT. The subtask remains in the WAIT until it is posted by the PNET driver to perform the disabling of the SCIP exit and to close down the VTAM interface.

Disabling the VTAM SCIP Exit: The VTAM SCIP Exit is quiesced when no more session requests originating from remote nodes can be accepted, for example after a PEND has been issued by the operator.

The VTAM SETLOGON QUIESCE request is issued and the subtask IPW$$S1 enters a wait state again until it is posted in order to perform the CLOSE request.

If VSE/POWER could not quiesce the VTAM SCIP exit, i. e. session requests issued by remote nodes can still be received, error message 1RD4I is issued including the reason codes causing the failure. The reason codes are taken from the VTAM RPL RTNCD and FDBK2 fields. The VTAM interface is closed and the PNET driver ECB is posted and the subtask is detached from VSE.

Closing the VTAM Interface: The subtask IPW$$S1 is posted by the PNET driver in order to close down the VTAM interface by means of the VTAM CLOSE macro. Message 1RE1I is issued after successful completion of the CLOSE request indicating that no more PNET SNA functions can be performed.

If the VTAM interface could not be closed properly error message 1RD5I (including the reason code) is issued. The reason code is taken from the VTAM ACB Error Field. The PNET driver ECB as well as the VSE/POWER Master ECB are posted and the subtask is detached from VSE.

PNET SNA Session Establishment

The IPW$$S2 routine is invoked in order to establish a session between the local and the remote node either on behalf of a primary or of a secondary application program. It is attached as a VSE/POWER task by the PNET driver.

The IPW$$S2 routine performs following functions:

1. The connect task establishes a session on behalf of a primary application program by issuing the VTAM OPNDST request in order to send a BIND command to the remote application program.
2. The connect task establishes a session on behalf of a secondary application program by issuing the VTAM OPNSEC request in order to accept a session request issued by the primary application program.
3. The connect task receives the session request from the primary application program but does not wish to accept it, so it issues the VTAM SESSIONC request.

Primary Application Program Establishes a Session: This routine is entered when the operator enters a PSTART PNET,nodeid command. The general logic flow is shown in the case of the first PSTART in Figure 74 on page 254 and in the case of subsequent PSTART commands in Figure 75 on page 255.

The command processor routine (IPW$$CM) builds a Node Control Block (NCB) which is passed to the PNET driver. The PNET driver attaches the connect task and passes the NCB address in R6 to it.

The connect task checks whether a session has already been established or is being established (caused by the session request of a remote node) and if so the PSTART command is ignored (without error message).

The SNA Session Control Block (SSCB) which contains all necessary VTAM control blocks as well as save areas and I/O areas as follows is then built.

The contents of the SSCB are as follows:

Save Areas	used for VTAM requests by connect task and the SEND/RECEIVE module IPW$$SR.
VTAM RPL	used for all VTAM requests by IPW$$S2 and IPW$$S3 as well as by all other PNET modules. (Except for SESSIONC.)
VTAM NIB	used for all VTAM requests by IPW$$S2 and IPW$$S3 as well as by all other PNET modules. (Except for SESSIONC.)
BIND image	used by connect task for OPNDST.
I/O Areas	used by IPW$$S2 for sending and receiving the NJE Type 4 FM Header.

The IPW$$S2 routine will wait for storage in the case that no storage is available.

After building the SSCB, the VTAM OPNDST is issued. A BIND command is sent to the secondary application program and is handled as described below.

The session is established after successful completion of the OPNDST request.

Both nodes have to agree on the buffer size used during data transfer between the two nodes and the process is as follows:

- The node with the higher node name sends the NJE Type 4 FM Header which contains the buffer size as defined in the NCB.
- The node with the lower node name receives the NJE Type 4 FM Header and sends its own FM Header including the buffer size as defined in its NCB.
- The node with higher node name receives the NJE Type 4 FM Header.
- Both nodes compare their own buffer size with the remote buffer size and the smaller of the two values is used for data transfer between the two nodes.

The connect task is detached after successful exchange of the buffer sizes.

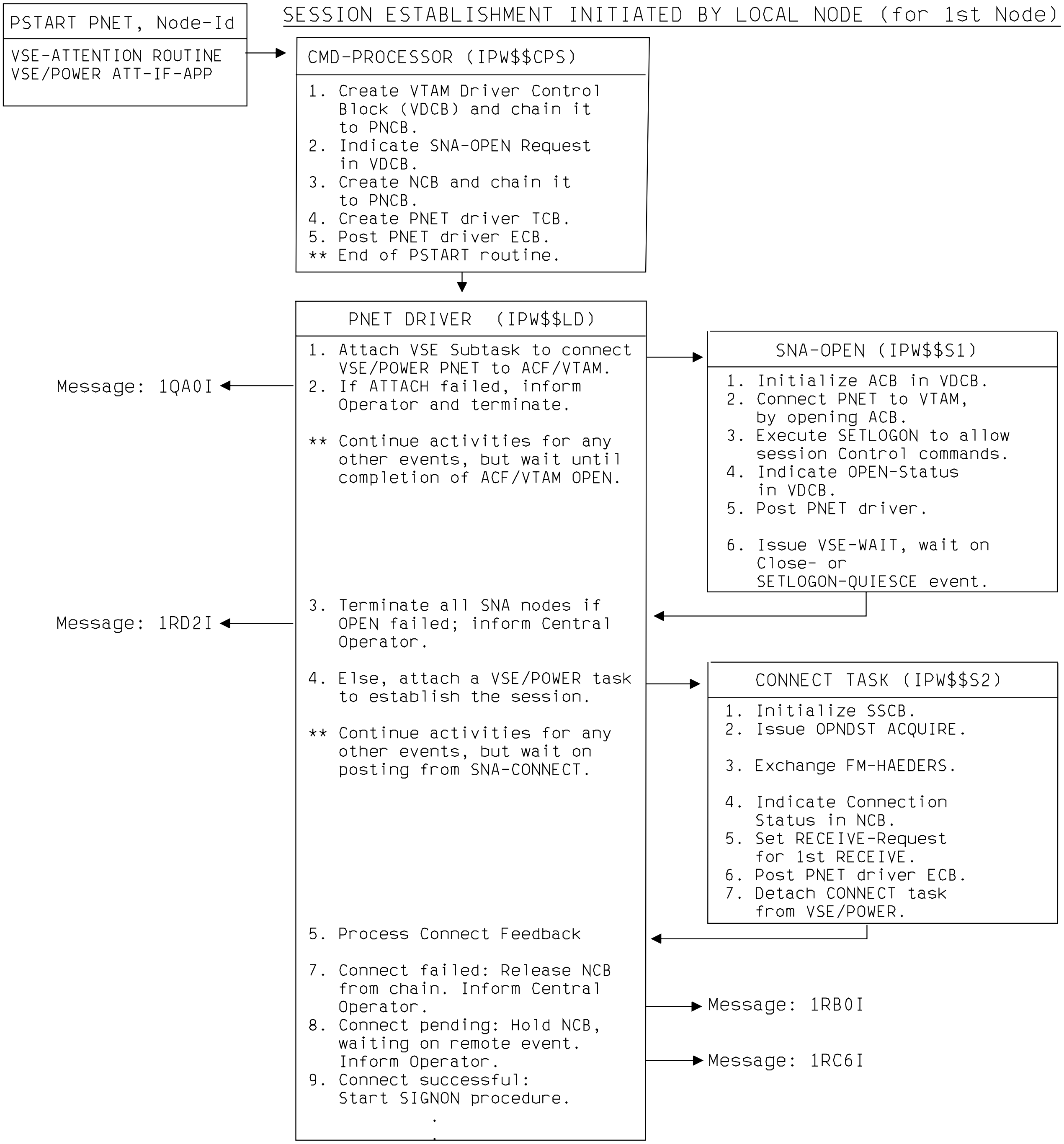

SESSION ESTABLISHMENT INITIATED BY LOCAL NODE (for 1st Node)
PSTART PNET, Node-Id
VSE-ATTENTION ROUTINE
VSE/POWER ATT-IF-APP
CMD-PROCESSOR (IPW$$CPS)
1. Create VTAM Driver Control Block (VDCB) and chain it to PNCB.
2. Indicate SNA-OPEN Request in VDCB.
3. Create NCB and chain it to PNCB.
4. Create PNET driver TCB.
5. Post PNET driver ECB.
** End of PSTART routine.
PNET DRIVER (IPW$$LD)
1. Attach VSE Subtask to connect VSE/POWER PNET to ACF/VTAM.
2. If ATTACH failed, inform Operator and terminate.
** Continue activities for any other events, but wait until completion of ACF/VTAM OPEN.
Message: 1QA0I
SNA-OPEN (IPW$$S1)
1. Initialize ACB in VDCB.
2. Connect PNET to VTAM, by opening ACB.
3. Execute SETLOGON to allow session Control commands.
4. Indicate OPEN-Status in VDCB.
5. Post PNET driver.
6. Issue VSE-WAIT, wait on Close- or SETLOGON-QUIESCE event.
3. Terminate all SNA nodes if OPEN failed; inform Central Operator.
Message: 1RD2I
4. Else, attach a VSE/POWER task to establish the session.
** Continue activities for any other events, but wait on posting from SNA-CONNECT.
CONNECT TASK (IPW$$S2)
1. Initialize SSCB.
2. Issue OPNDST ACQUIRE.
3. Exchange FM-HAEDERS.
4. Indicate Connection Status in NCB.
5. Set RECEIVE-Request for 1st RECEIVE.
6. Post PNET driver ECB.
7. Detach CONNECT task from VSE/POWER.
5. Process Connect Feedback
7. Connect failed: Release NCB from chain. Inform Central Operator.
Message: 1RB0I
8. Connect pending: Hold NCB, waiting on remote event. Inform Operator.
Message: 1RC6I
9. Connect successful: Start SIGNON procedure.
.
.

Figure 74. PNET SNA Session Establishment - for First Node

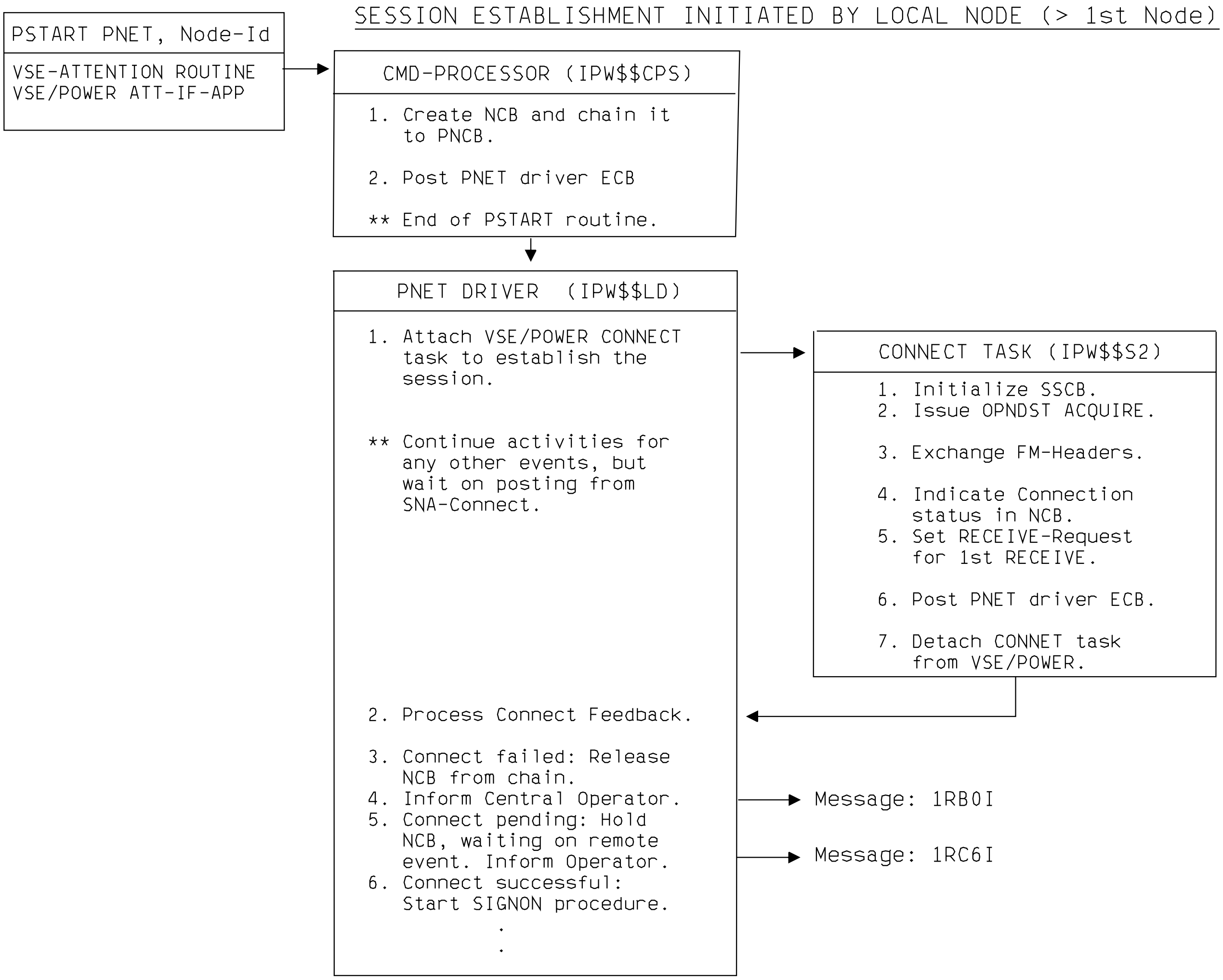

SESSION ESTABLISHMENT INITIATED BY LOCAL NODE (> 1st Node)
PSTART PNET, Node-Id
VSE-ATTENTION ROUTINE
VSE/POWER ATT-IF-APP
CMD-PROCESSOR (IPW$$CPS)
1. Create NCB and chain it to PNCB.
2. Post PNET driver ECB
** End of PSTART routine.
PNET DRIVER (IPW$$LD)
1. Attach VSE/POWER CONNECT task to establish the session.
** Continue activities for any other events, but wait on posting from SNA-Connect.
CONNECT TASK (IPW$$S2)
1. Initialize SSCB.
2. Issue OPNDST ACQUIRE.
3. Exchange FM-Headers.
4. Indicate Connection status in NCB.
5. Set RECEIVE-Request for 1st RECEIVE.
6. Post PNET driver ECB.
7. Detach CONNET task from VSE/POWER.
2. Process Connect Feedback.
3. Connect failed: Release NCB from chain.
4. Inform Central Operator.
5. Connect pending: Hold NCB, waiting on remote event. Inform Operator.
6. Connect successful: Start SIGNON procedure.
.
.
Message: 1RB0I
Message: 1RC6I

Figure 75. PNET SNA Session Establishment. For non-first node.

VTAM OPNDST Error: Two cases have to be distinguished:

1. The NCB will be kept, which means that the local PNET system will be able to receive a session request from this node. Error message 1RD8I is displayed indicating the reason code and message 1RC6I is displayed indicating that the local node remains in a pending status.
2. The NCB will not be kept, which means that an unrecoverable error has occurred. Error message 1RD8I is displayed indicating the reason of the OPNDST failure.

SEND/RECEIVE Error During FM Header Exchange: Two cases have to be distinguished:

1. An VTAM error has occurred which means that the exchange of the FM Headers has not been successful. Error message 1RD8I is displayed indicating the reason code and the session has to be terminated by the disconnect task.
2. Invalid data (incorrect or no FM Header) has been received. Error message 1RD8I is displayed indicating the reason of the failure. The session has to be terminated by the disconnect task.

Secondary Application Program Accepts Session Request: This routine is entered if the operator at the remote node has entered a PSTART PNET,nodeid command or an equivalent command from another supported system. The general logic flow for this case is shown in Figure 76 on page 257.

The PNET driver attaches the connect task and passes the SRQE address in register 6.

The connect task checks whether a session has already been established or is being established (caused by a PSTART command at the local console) and if so the session request is ignored (without error message).

The SNA Session Control Block (SSCB) is built as described above.

After building the SSCB, the VTAM OPNSEC is issued. This means a positive response to the BIND command is sent to the primary application program.

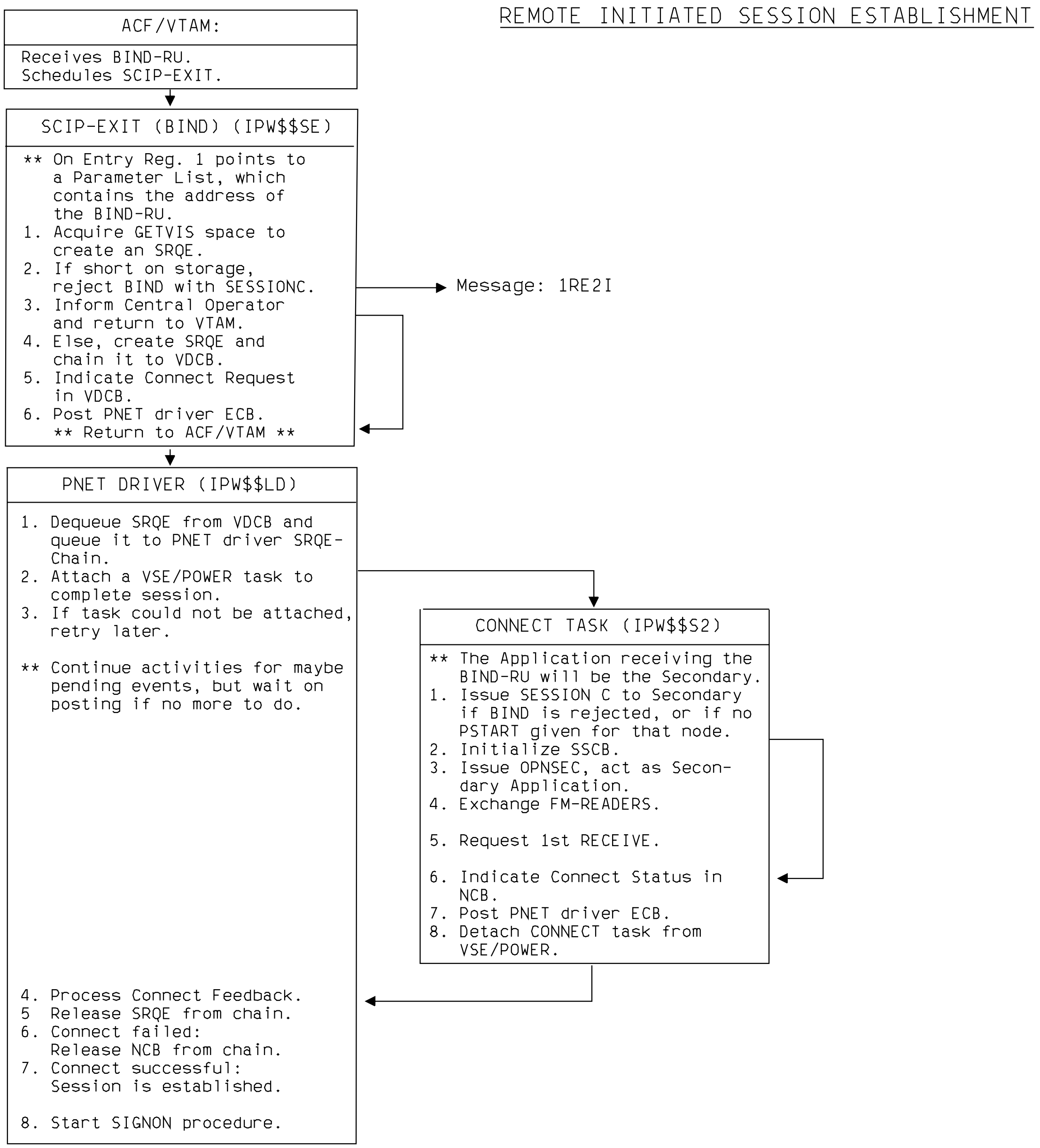
The session is established after successful completion of the OPNSEC request.

Both nodes have to agree on the buffer size used during data transfer between the two nodes. The same mechanism is used as described above.

The connect task is detached after successful exchange of the buffer sizes.

If an error occurred while processing the OPNSEC macro, the NCB is not kept which means that an unrecoverable error has occurred. Error message 1RD8I is displayed indicating the reason of the OPNSEC failure.

A VARY NET,INACT,SID=applid,TYPE=FORCE should be entered in order to terminate the session.



REMOTE INITIATED SESSION ESTABLISHMENT

```
ACF/VTAM:

Receives BIND-RU.
Schedules SCIP-EXIT.
```

```
SCIP-EXIT (BIND) (IPW$$SE)

** On Entry Reg. 1 points to
   a Parameter List, which
   contains the address of
   the BIND-RU.
1. Acquire GETVIS space to
   create an SRQE.
2. If short on storage,
   reject BIND with SESSIONC.
3. Inform Central Operator
   and return to VTAM.
4. Else, create SRQE and
   chain it to VDCB.
5. Indicate Connect Request
   in VDCB.
6. Post PNET driver ECB.
   ** Return to ACF/VTAM **
```

Message: 1RE2I

```
PNET DRIVER (IPW$$LD)

1. Dequeue SRQE from VDCB and
   queue it to PNET driver SRQE-
   Chain.
2. Attach a VSE/POWER task to
   complete session.
3. If task could not be attached,
   retry later.

** Continue activities for maybe
   pending events, but wait on
   posting if no more to do.

4. Process Connect Feedback.
5  Release SRQE from chain.
6. Connect failed:
   Release NCB from chain.
7. Connect successful:
   Session is established.

8. Start SIGNON procedure.
```

```
CONNECT TASK (IPW$$S2)

** The Application receiving the
   BIND-RU will be the Secondary.
1. Issue SESSION C to Secondary
   if BIND is rejected, or if no
   PSTART given for that node.
2. Initialize SSCB.
3. Issue OPNSEC, act as Secon-
   dary Application.
4. Exchange FM-READERS.

5. Request 1st RECEIVE.

6. Indicate Connect Status in
   NCB.
7. Post PNET driver ECB.
8. Detach CONNECT task from
   VSE/POWER.
```

Figure 76. PNET SNA Remote Initiated Session

Secondary Application Program Rejects Session Request: The operator at the remote console has entered the

```
PSTART PNET,nodeid[,nodepassword]
```

command.

The VTAM SCIP Exit routine (IPW$$SE) has been scheduled with the indication that a BIND command has been received. The IPW$$SE routine builds a Session Request Element (SRQE) which is passed to the PNET driver. The PNET driver attaches the connect task and passes the SRQE address in register 6 to it.

The IPW$$S2 routine checks whether the session request represented by the SRQE can be accepted or not.

The session request is rejected when one of the following situations are encountered:

1. The APPLID contained in the BIND image (which is contained in the SRQE) is not defined in the local network definition table (NDT). Error message 1RD6I is displayed at the local console.
2. The local operator has not yet entered a PSTART PNET,nodeid command for the remote node. Error message 1RC7I is displayed at the local console.
3. The BIND image which has been received contains specifications (bytes 0-2) which do not agree with the specifications PNET is using. Error message 1RE2I is displayed at the local console.
4. The local operator has entered a PSTOP command for the remote node or VTAM has terminated or is going to terminate. No error message is displayed at the local console.
5. A primary session is being established or has already been established. No error message is displayed at the local console.
6. No virtual storage could be acquired to build the required VTAM control blocks. No error message is displayed at the local console.

An VTAM SESSIONC request is issued in order to send a negative reply to the OPNDST issued by the primary application program located at the remote node.

The necessary VTAM RPL and NIB control blocks are contained in the SRQE itself and have been prepared by the VTAM SCIP Exit routine IPW$$SE.

System sense information as well as system sense modifier information are sent to the primary application program to inform it about the reason for the session request rejection.

The following sense code are set (always SENSEO=RR (X'08')):

SSENSMO=X'05' Session limit exceeded.

SSENSMO=X'0F' End user not authorized.

SSENSMO=X'12' Insufficient resources.

SSENSMO=X'15' Function already active.

SSENSMO=X'21' Invalid session parameters.

This system sense information is displayed at the remote console with error message 1RD8I.

The connect task detaches itself from VSE/POWER after successful completion of the SESSIONC request. No session has been established.

Error message 1RD8I is displayed in case of SESSIONC failure. The still outstanding session request should be terminated by entering VARY NET,INACT,SID=nnnnnnnn,TYPE=FORCE.

PNET SNA Session Termination

The IPW$$S3 routine is invoked in order to terminate a session which has been established by the connect task. It is attached as a VSE/POWER task by the PNET driver.

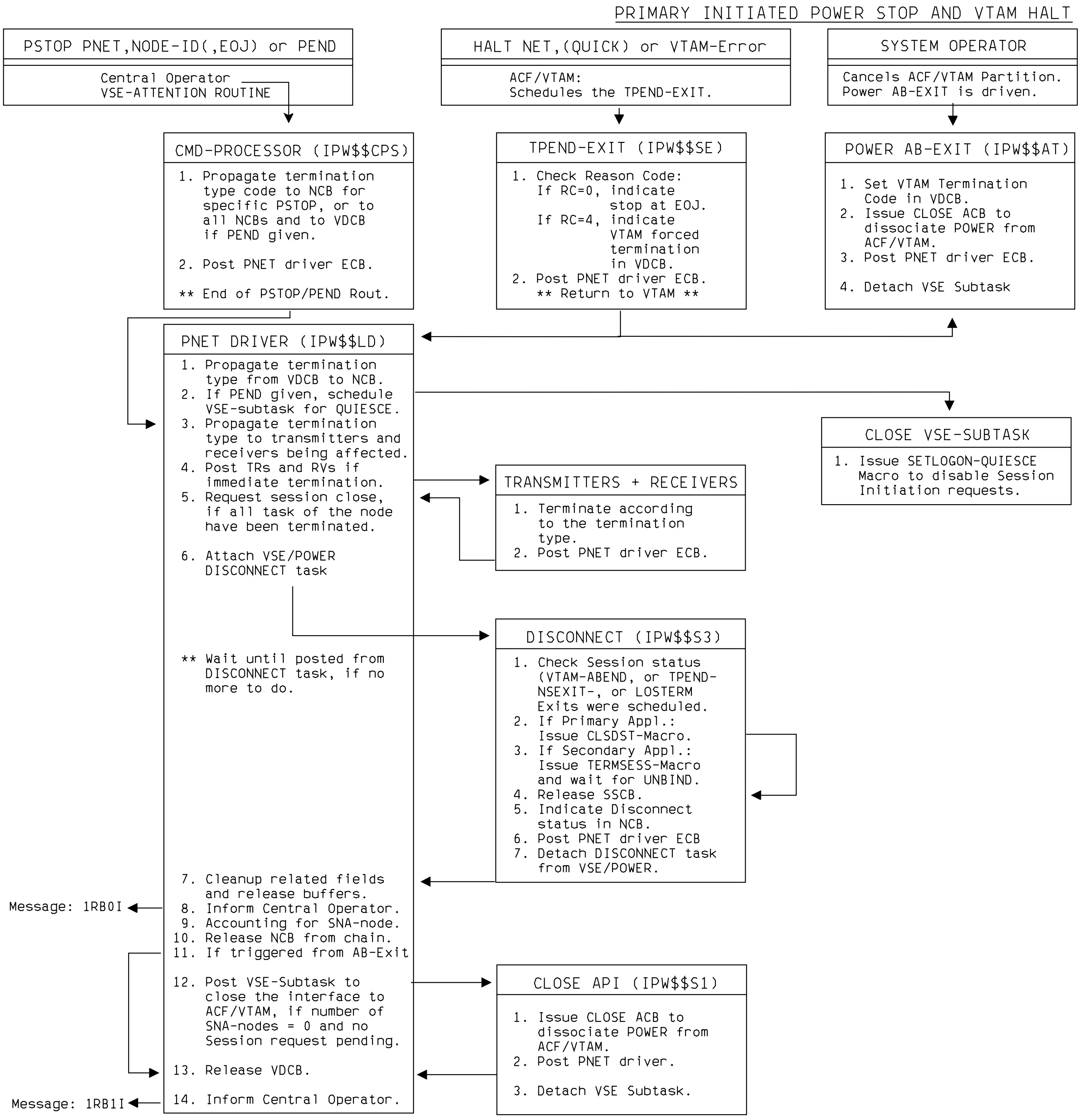
The IPW$$S3 routine performs following functions:

1. Terminates a session in behalf of a primary application program by issuing the VTAM CLSDST request.
2. Terminates a session in behalf of a secondary application program by issuing the VTAM TERMSESS request.
3. Receives the session termination request from the primary application program and acts on behalf of a secondary application program. The IPW$$S3 routine receives the UNBIND command.
4. Receives the session termination request from the secondary application program and acts on behalf of a primary application program. The IPW$$S3 routine receives the LOSTERM RC=20 condition.

Primary Application Program Terminates the Session: The operator at the local console enters a PSTOP PNET,nodeid command or another condition (like VTAM Halt) leads to a termination request. The general logic flow is shown in Figure 77 on page 260.

The PNET driver attaches the disconnect task which then waits for the completion of the connect task which has established or which is just establishing the session. The necessary ECB is located in the NCB representing the session.

The VTAM request CLSDST is issued to disconnect the application programs located on the local and remote node. The CLSDST request causes an UNBIND command to be sent to the remote secondary application program (via the VTAM SCIP Exit) thus informing the secondary application program about session termination. The NCB is flagged to indicate that it can be freed by the PNET driver. The PNET driver ECB is posted and the disconnect task detaches itself from VSE/POWER. The SNA Session Control Block (SSCB) is freed later by the PNET driver.



PRIMARY INITIATED POWER STOP AND VTAM HALT
PSTOP PNET,NODE-ID(,EOJ) or PEND
Central Operator
VSE-ATTENTION ROUTINE
HALT NET,(QUICK) or VTAM-Error
ACF/VTAM:
Schedules the TPEND-EXIT.
SYSTEM OPERATOR
Cancels ACF/VTAM Partition.
Power AB-EXIT is driven.
Message: 1RB0I
Message: 1RB1I

CMD-PROCESSOR (IPW$$CPS)

1. Propagate termination type code to NCB for specific PSTOP, or to all NCBs and to VDCB if PEND given.
2. Post PNET driver ECB.

** End of PSTOP/PEND Rout.

TPEND-EXIT (IPW$$SE)

1. Check Reason Code:
   If RC=0, indicate stop at EOJ.
   If RC=4, indicate VTAM forced termination in VDCB.
2. Post PNET driver ECB.
   ** Return to VTAM **

POWER AB-EXIT (IPW$$AT)

1. Set VTAM Termination Code in VDCB.
2. Issue CLOSE ACB to dissociate POWER from ACF/VTAM.
3. Post PNET driver ECB.
4. Detach VSE Subtask

PNET DRIVER (IPW$$LD)

1. Propagate termination type from VDCB to NCB.
2. If PEND given, schedule VSE-subtask for QUIESCE.
3. Propagate termination type to transmitters and receivers being affected.
4. Post TRs and RVs if immediate termination.
5. Request session close, if all task of the node have been terminated.
6. Attach VSE/POWER DISCONNECT task

** Wait until posted from DISCONNECT task, if no more to do.

7. Cleanup related fields and release buffers.
8. Inform Central Operator.
9. Accounting for SNA-node.
10. Release NCB from chain.
11. If triggered from AB-Exit
12. Post VSE-Subtask to close the interface to ACF/VTAM, if number of SNA-nodes = 0 and no Session request pending.
13. Release VDCB.
14. Inform Central Operator.

CLOSE VSE-SUBTASK

1. Issue SETLOGON-QUIESCE Macro to disable Session Initiation requests.

TRANSMITTERS + RECEIVERS

1. Terminate according to the termination type.
2. Post PNET driver ECB.

DISCONNECT (IPW$$S3)

1. Check Session status (VTAM-ABEND, or TPEND-NSEXIT-, or LOSTERM Exits were scheduled.
2. If Primary Appl.: Issue CLSDST-Macro.
3. If Secondary Appl.: Issue TERMSESS-Macro and wait for UNBIND.
4. Release SSCB.
5. Indicate Disconnect status in NCB.
6. Post PNET driver ECB
7. Detach DISCONNECT task from VSE/POWER.

CLOSE API (IPW$$S1)

1. Issue CLOSE ACB to dissociate POWER from ACF/VTAM.
2. Post PNET driver.
3. Detach VSE Subtask.

Figure 77. PNET SNA Primary Initiated Stop

Secondary Application Program Terminates the Session: The operator at the local console enters the PSTOP PNET,nodeid command or another condition (like VTAM Halt) leads to a termination request.

The PNET driver attaches the disconnect task which then waits for the completion of the connect task which has established or which is just establishing the session. The necessary ECB is located in the NCB representing the session.

The VTAM request TERMSESS is issued to request session termination from the primary application program. The TERMSESS request causes the VTAM LOSTERM Exit to be scheduled for the remote primary application program (via the VTAM LOSTERM Exit with Reason Code 20) thus informing the primary application program about the session termination request.

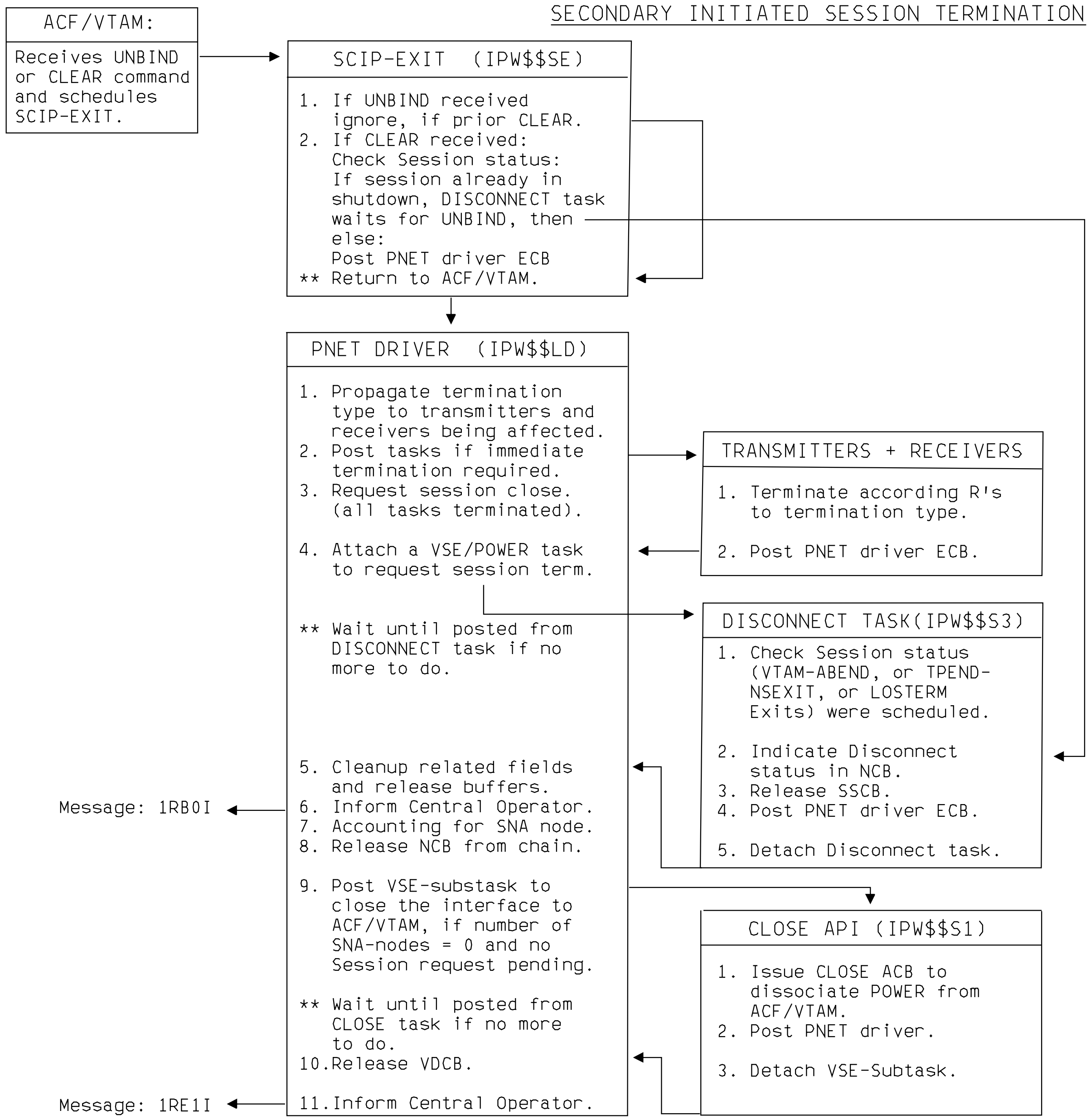
The primary application program has to issue an VTAM CLSDST request so that the session is terminated. The secondary application program receives an UNBIND command for which it has waited after issuing the TERMSESS request.

The NCB is flagged that it can be freed by the PNET driver. The PNET driver ECB is posted and the disconnect task detaches itself from VSE/POWER. The SNA Session Control Block (SSCB) is freed later by the PNET driver.

Primary Application Program Receives Session Termination Request: The operator at the remote console enters PSTOP PNET,nodeid command or another condition (like VTAM Halt) leads to a termination request. The general flow is shown in Figure 78 on page 262.

The PNET driver attaches the disconnect task which then waits for the completion of the connect task which has established or which is just establishing the session. The necessary ECB is located in the NCB representing the session.

LOSTERM condition with reason code 20 has been passed to PNET through the VTAM LOSTERM Exit. A CLSDST request (as described above) has to be issued in order to terminate properly the session. The CLSDST request causes an UNBIND command to be sent to the remote secondary application program (via the VTAM SCIP Exit) thus informing the secondary application program about session termination. The NCB is flagged that it can be freed by the PNET driver. The PNET driver ECB is posted and the disconnect task detaches itself from VSE/POWER. The SNA Session Control Block (SSCB) is freed later by the PNET driver.



SECONDARY INITIATED SESSION TERMINATION

ACF/VTAM:

Receives UNBIND or CLEAR command and schedules SCIP-EXIT.

SCIP-EXIT (IPW$$SE)

1. If UNBIND received ignore, if prior CLEAR.
2. If CLEAR received:
   Check Session status:
   If session already in shutdown, DISCONNECT task waits for UNBIND, then
   else:
   Post PNET driver ECB

** Return to ACF/VTAM.

PNET DRIVER (IPW$$LD)

1. Propagate termination type to transmitters and receivers being affected.
2. Post tasks if immediate termination required.
3. Request session close. (all tasks terminated).
4. Attach a VSE/POWER task to request session term.

** Wait until posted from DISCONNECT task if no more to do.

5. Cleanup related fields and release buffers.
6. Inform Central Operator.
7. Accounting for SNA node.
8. Release NCB from chain.
9. Post VSE-substask to close the interface to ACF/VTAM, if number of SNA-nodes = 0 and no Session request pending.

** Wait until posted from CLOSE task if no more to do.

10. Release VDCB.
11. Inform Central Operator.

Message: 1RB0I

Message: 1RE1I

TRANSMITTERS + RECEIVERS

1. Terminate according R's to termination type.
2. Post PNET driver ECB.

DISCONNECT TASK(IPW$$S3)

1. Check Session status (VTAM-ABEND, or TPEND-NSEXIT, or LOSTERM Exits) were scheduled.
2. Indicate Disconnect status in NCB.
3. Release SSCB.
4. Post PNET driver ECB.
5. Detach Disconnect task.

CLOSE API (IPW$$S1)

1. Issue CLOSE ACB to dissociate POWER from ACF/VTAM.
2. Post PNET driver.
3. Detach VSE-Subtask.

Figure 78. PNET SNA Secondary Initiated Stop

Secondary Application Program Receives Session Termination Request: The operator at the remote console enters the

```
PSTOP PNET,nodeid
```

command or another condition (like VTAM Halt) leads to a termination request. The PNET driver attaches the disconnect task which then waits for the completion of the connect task which has established or which is just establishing the session. The necessary ECB is located in the NCB representing the session.

The VTAM request CLSDST has been issued by the primary application program. This request causes an UNBIND command to be sent to the remote secondary application program (via the VTAM SCIP Exit) thus informing the secondary application program about session termination. The NCB is flagged that it can be freed by the PNET driver. The PNET driver ECB is posted and the disconnect task detaches itself from VSE/AF. The SNA Session Control Block (SSCB) is freed later by the PNET driver.

In case an VTAM error has been encountered during execution of the CLSDST or TERMSESS request, error message 1RD8I is issued including the reason code (which consists of the RPL RTNCD and FDBK2 codes. Error message 1RC4I will be displayed when VTAM suffers from temporary storage shortage and cannot perform the requested function.

The console operator should try to cancel the still existing session by using the VARY NET,INACT,SID=xxxxxxxx,TYPE=FORCE.

PNET SNA VTAM Exits

The module IPW$$SE contains the VTAM Exit Routines for the PNET SNA Support. The following exits are supported:

SCIP
NSEXIT
LOSTERM
TPEND

The exits are required to handle the special events scheduled by VTAM.

The exit routines run under control of the VSE/AF subtask under which the VTAM OPEN was performed. During OPEN time the exits are enabled to VTAM via the SETLOGON command issued by module IPW$$S1.

Because the exits run under the control of the VSE/AF subtask and cannot share VSE/POWER resources, all events which cause an exit to be scheduled are passed to the PNET driver by setting various indications in control blocks. The following exit routines are supported and perform the following functions.

SCIP Exit: The SCIP exit is scheduled in the following circumstances:

- BIND Request Unit

  As part of the session establishment an OPNDST command is executed which triggers the sending of a BIND-RU to the other end of the session. The received BIND-RU causes VTAM to schedule the SCIP-Exit with a parameter list containing the address of the received BIND-RU.

  The BIND-RU is used to form a Session Request Element (SRQE) in the GETVIS space of the VSE/POWER partition. The SRQE is chained to the PNET master control block (PNCB) and the PNET driver is informed that a remote session request has arrived.

If no GETVIS storage is available the BIND-RU is rejected via a SESSIONC command which triggers the completion of the OPNDST with error. The operator is informed that the session request has been rejected.

- UNBIND Command

  As part of the session termination process the primary application program executes the CLSDST command, which triggers VTAM to send an UNBIND command to the secondary application program.

  If the secondary initiates the termination and executes the TERMSESS command, VTAM on the primary side triggers the LOSTERM exit but sends an UNBIND command back to the secondary end.

  In both cases, a received UNBIND command forces VTAM to schedule the SCIP Exit with a read-only RPL, which contains, in its session control field, the indication 'UNBIND received'. The exit propagates this event to the PNET driver to stop session communication.

- Start Data Traffic (SDT) Command

  At the beginning of a session, an SDT command is sent from the primary end to inform the secondary end that flow of data requests, data flow control commands, and responses may be started. The exit informs the connect task, which is waiting on this event, that data traffic can now be started.

- CLEAR Command

  The CLEAR command is sent by the primary application when the flow of data requests, data flow commands, and responses is to be stopped, either because the primary application is terminating or needs to take some recovery action.

  A CLEAR command forces VTAM to schedule the SCIP exit with a read-only RPL which contains, in its session control field, the indication 'CLEAR received'. The exit propagates this event to the PNET driver to stop the session communication.

- Other Session Control Commands

  RQR and STSN As part of the session recovery, these commands are used to cleanup the session. Both commands are not supported by VSE/POWER but when received from other components they lead to termination of the session.

  DFASY Asynchronous data flow request are handled by the PNET driver via the receive function.

LOSTERM Exit: Various situations lead to the triggering of the LOSTERM exit. VTAM schedules the exit by identifying the situations via the reason lost code, which is passed in the parameter list.

The following situations cause the LOSTERM exit to be scheduled:

20 The secondary end issued a TERMSESS... TYPE=UNCOND command to terminate the session. VTAM at the primary side schedules the exit. In this case, the exit informs the PNET driver to terminate the session immediately.

32 Network operator initiated conditional terminate.

This event causes the PNET driver to be informed that a conditional terminate should take place, which implies that all transmitter and receiver tasks may continue until end-of-job, before the session is terminated.

12 Network operator initiated VTAM HALT.

This event causes an immediate termination of VTAM. The PNET driver is informed that VTAM is terminating immediately and propagates an immediate termination to all PNET SNA sessions.

36 VTAM buffer limit exceeded.

If the buffer limit defined by the NCP Generation is exceeded, the exit is scheduled. This event causes immediate termination of the session and the PNET driver is informed about this event.

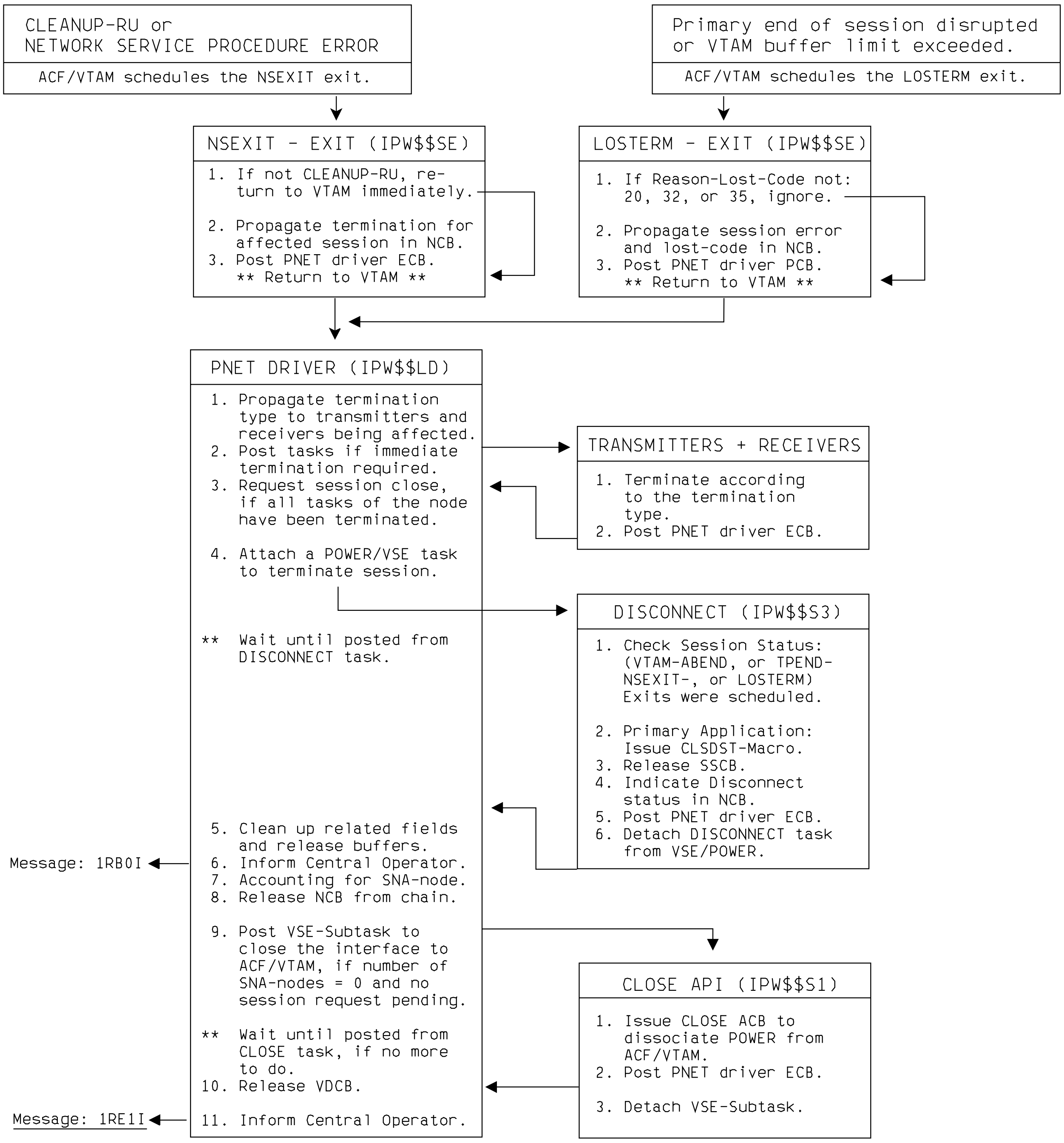
The operator is informed with message 1RD7I that the LOSTERM exit has been scheduled.

NSEXIT Exit: The CLEANUP-RU is the only type of RU which is supported in this exit. A CLEANUP-RU is created by VTAM under the following circumstances: The general flow is shown in Figure 79 on page 266.

Operator initiated command:
VARY NET,INACT.....
VARY NET,TERM,SID=nnnnnn,TYPE=FORCE or
Unexpected CLOSE issued from application.

The received CLEANUP-RU leads to an immediate termination of the session. The exit informs the PNET driver about this situation.

TERMINATION FORCED BY NETWORK ERRORS



CLEANUP-RU or NETWORK SERVICE PROCEDURE ERROR

ACF/VTAM schedules the NSEXIT exit.

Primary end of session disrupted or VTAM buffer limit exceeded.

ACF/VTAM schedules the LOSTERM exit.

NSEXIT - EXIT (IPW$$SE)

1. If not CLEANUP-RU, return to VTAM immediately.
2. Propagate termination for affected session in NCB.
3. Post PNET driver ECB.

** Return to VTAM **

LOSTERM - EXIT (IPW$$SE)

1. If Reason-Lost-Code not: 20, 32, or 35, ignore.
2. Propagate session error and lost-code in NCB.
3. Post PNET driver PCB.

** Return to VTAM **

PNET DRIVER (IPW$$LD)

1. Propagate termination type to transmitters and receivers being affected.
2. Post tasks if immediate termination required.
3. Request session close, if all tasks of the node have been terminated.
4. Attach a POWER/VSE task to terminate session.

** Wait until posted from DISCONNECT task.

5. Clean up related fields and release buffers.
6. Inform Central Operator.
7. Accounting for SNA-node.
8. Release NCB from chain.
9. Post VSE-Subtask to close the interface to ACF/VTAM, if number of SNA-nodes = 0 and no session request pending.

** Wait until posted from CLOSE task, if no more to do.

10. Release VDCB.
11. Inform Central Operator.

Message: 1RB0I

Message: 1RE1I

TRANSMITTERS + RECEIVERS

1. Terminate according to the termination type.
2. Post PNET driver ECB.

DISCONNECT (IPW$$S3)

1. Check Session Status: (VTAM-ABEND, or TPEND-NSEXIT-, or LOSTERM) Exits were scheduled.
2. Primary Application: Issue CLSDST-Macro.
3. Release SSCB.
4. Indicate Disconnect status in NCB.
5. Post PNET driver ECB.
6. Detach DISCONNECT task from VSE/POWER.

CLOSE API (IPW$$S1)

1. Issue CLOSE ACB to dissociate POWER from ACF/VTAM.
2. Post PNET driver ECB.
3. Detach VSE-Subtask.

Figure 79. PNET SNA Abnormal Termination

TPEND Exit: VTAM schedules the TPEND exit if the network operator is halting VTAM via the HALT command. Three conditions which inform the application are supported:

```
HALT NET        (normal shutdown)
HALT NET,QUICK  (Immediate termination)
VTAM ABNORMAL TERMINATION
```

VTAM schedules the TPEND exit with a reason code as defined below.

0 HALT NET was issued to shutdown VTAM in a normal fashion. The PNET driver is informed to shutdown all sessions in a normal way.

4 HALT NET,QUICK was issued to terminate VTAM immediately. The PNET driver is informed to terminate all sessions without waiting for end-of-job conditions.

8 VTAM ABNORMAL TERMINATION has taken place. The PNET driver is informed to terminate all sessions without waiting for end-of-job conditions.

PNET SNA SEND/RECEIVE Function

The SEND and RECEIVE functions of the PNET-SNA support are located in the module IPW$$SR. This module also contains the SEND/RECEIVE exit routines for asynchronous processing. The functions are called either:

- By the PNET driver, in which case the routines run under control of the calling task TCB acting as a function module.
  - The PNET driver calls the SEND function by means of the IPW$IOM macro instruction when at least one output buffer has been queued by the PNET driver into the 'TO-BE-SENT AHEAD' queue and no SEND is currently in progress.
  - The PNET driver calls the RECEIVE function by means of the IPW$IOM macro when at least one input buffer has been queued by the PNET driver to the 'FREE INPUT AHEAD' queue and no RECEIVE is currently in progress.
- By the SEND/RECEIVE exit routines, which are scheduled by VTAM after final completion of the request, by means of the IPW$IOM macro instruction.

  Note: Both the exits as well as the invoked SEND/RECEIVE functions run under control of the PNET SNA subtask (IPW$$S1).

The following technique is used to prevent concurrent SEND/RECEIVE being executed:

If the SEND or RECEIVE functions are called from the PNET driver, the function gates are locked and remain locked until no more buffers are available to send or to receive requests or responses. Due to asynchronous processing, the SEND/RECEIVE exits are scheduled by VTAM at final completion time. The currently held input or output buffer is queued to the PNET driver 'BUFFER QUEUE' for later processing. The SEND or RECEIVE exit then initiates a new SEND or RECEIVE request by executing the IPW$IOM macro. The appropriate function gate remains locked unless an error is detected or no buffer can be sent or received.

Two additional queues are introduced, for performance reasons, in the SNA processing:

- To be sent ahead queue
- Free input ahead queue

The queues provide the capability to initiate another SEND/RECEIVE from the appropriate exit which runs under the TIK of the PNET SNA subtask. The normal input/output buffer queues are maintained by buffer service using the VSE/POWER resource management which is not available when running as VSE/AF subtask.

The 'ahead' queues are filled by the PNET driver on demand of the SEND/RECEIVE function invoked by the exits. Once a SEND/RECEIVE has been initiated by the PNET driver, the next SEND/RECEIVE is automatically issued by the exit when the previous has finally completed. This process continues until no buffers are available or an error has been encountered.

The following queues are used by the SNA processing routines:

TO-BE-SENT-AHEAD Queue This buffer queue is updated by the PNET driver when a 'SEND-REQUEST' is indicated in the appropriate node control block (NCB). The output buffers from the 'TO-BE-SENT QUEUE' are chained at the tail of the 'TO-BE-SENT-AHEAD' queue.

FREE-INPUT AHEAD Queue This buffer queue is updated by the PNET driver when a 'RECEIVE-REQUEST' is indicated in the NCB. The input buffers from the 'FREE INPUT QUEUE' are chained on top of the 'FREE-INPUT-AHEAD' queue.

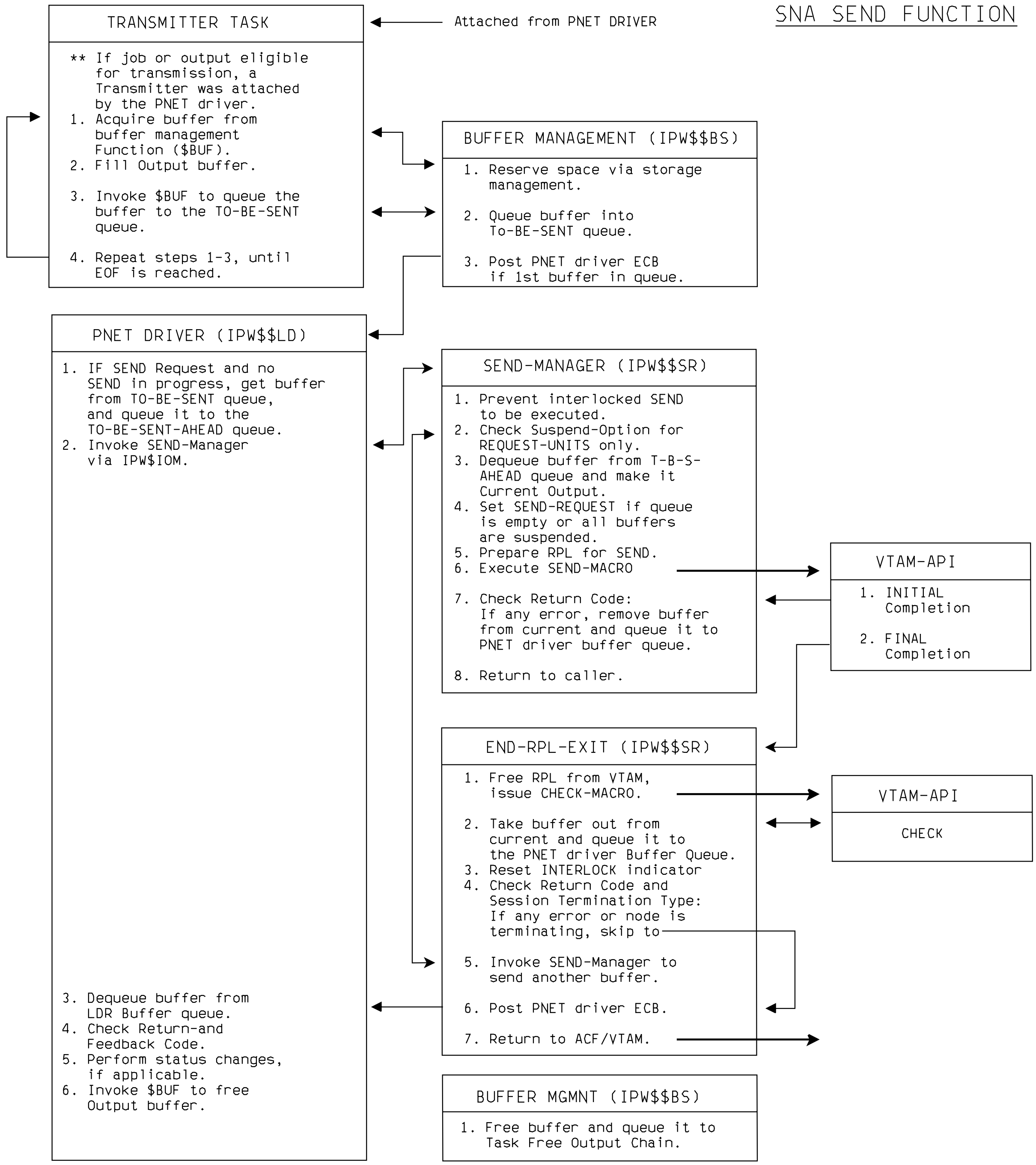
PNET driver BUFFER Queue This queue contains all input and output buffers, which were either sent or received by the SEND/RECEIVE function, for further processing by the PNET driver. At final completion time of a SEND/RECEIVE, or at initial completion time in the case of a failure, the buffer with its associated RPL is queued on top of the queue. The queue is anchored to the PNET driver task TCB and is re-ordered to the correct sequence (first-in, first-out) by the PNET driver.

SEND Function: The SEND-function checks the SEND-gate and returns to the caller with return code X'FF' in register 15 if the gate is already locked. Otherwise the gate is locked to prevent an additional SEND being executed and the TO-BE-SENT-AHEAD queue is scanned to find a buffer which is eligible to send. Each output buffer in turn is examined for eligibility for transmission. This is necessary because any receiver on the other node may temporarily stop transmission by sending a “suspend” control record. The stream can be resumed by sending a “resume” stream control record.

Output buffers may contain either Request-Units (Data-RUs) or Response-Units (Response-RUs). Response-RUs are always sent, even when the particular data-stream is suspended. If the output buffer is owned by the PNET driver, then that buffer is also sent. If no buffer is ready to send, the PNET driver is informed by setting the 'SEND-Request' in the NCB, the gate is opened and return is made to the caller. If, however, an output buffer is eligible for transmission, the RPL, contained in the buffer, is modified according to the data in the buffer and the SEND macro is executed. If no RPL is present, the pre-generated RPL of the SNA Session Control Block (SSCB) is used as a skeleton and modified accordingly.

The initial return code is checked and, if no error is indicated, return is made to the caller. If the initial completion fails, the output buffer causing the error is chained to the PNET driver buffer queue, the gate is unlocked, and return is made to the caller.

The flow in the SEND exit is shown in Figure 80 on page 269.



SNA SEND FUNCTION

TRANSMITTER TASK

Attached from PNET DRIVER

** If job or output eligible for transmission, a Transmitter was attached by the PNET driver.

1. Acquire buffer from buffer management Function ($BUF).
2. Fill Output buffer.
3. Invoke $BUF to queue the buffer to the TO-BE-SENT queue.
4. Repeat steps 1-3, until EOF is reached.

BUFFER MANAGEMENT (IPW$$BS)

1. Reserve space via storage management.
2. Queue buffer into To-BE-SENT queue.
3. Post PNET driver ECB if 1st buffer in queue.

PNET DRIVER (IPW$$LD)

1. IF SEND Request and no SEND in progress, get buffer from TO-BE-SENT queue, and queue it to the TO-BE-SENT-AHEAD queue.
2. Invoke SEND-Manager via IPW$IOM.
3. Dequeue buffer from LDR Buffer queue.
4. Check Return-and Feedback Code.
5. Perform status changes, if applicable.
6. Invoke $BUF to free Output buffer.

SEND-MANAGER (IPW$$SR)

1. Prevent interlocked SEND to be executed.
2. Check Suspend-Option for REQUEST-UNITS only.
3. Dequeue buffer from T-B-S-AHEAD queue and make it Current Output.
4. Set SEND-REQUEST if queue is empty or all buffers are suspended.
5. Prepare RPL for SEND.
6. Execute SEND-MACRO
7. Check Return Code: If any error, remove buffer from current and queue it to PNET driver buffer queue.
8. Return to caller.

VTAM-API

1. INITIAL Completion
2. FINAL Completion

END-RPL-EXIT (IPW$$SR)

1. Free RPL from VTAM, issue CHECK-MACRO.
2. Take buffer out from current and queue it to the PNET driver Buffer Queue.
3. Reset INTERLOCK indicator
4. Check Return Code and Session Termination Type: If any error or node is terminating, skip to
5. Invoke SEND-Manager to send another buffer.
6. Post PNET driver ECB.
7. Return to ACF/VTAM.

VTAM-API

CHECK

BUFFER MGMNT (IPW$$BS)

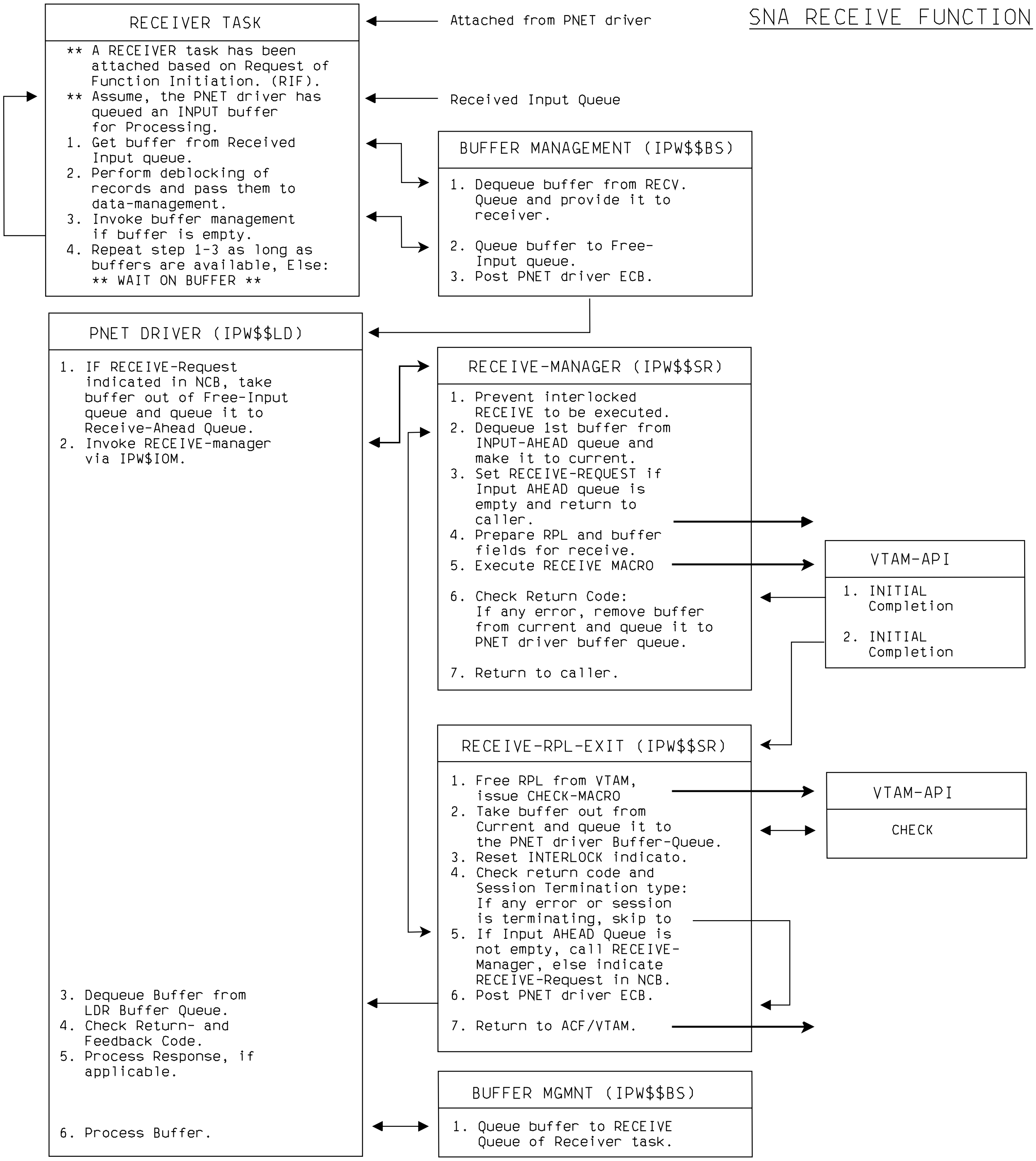
1. Free buffer and queue it to Task Free Output Chain.

Figure 80. PNET SEND Function

RECEIVE Function: The RECEIVE-function checks the RECEIVE-gate and returns to the caller with return code X'FF' in register 15 if the gate is already locked. Otherwise the gate is locked to prevent more than one RECEIVE being executed at a time. The FREE-INPUT AHEAD queue is examined for a free buffer. If no buffer is available, the PNET driver is informed by setting the 'RECEIVE-Request' in the NCB, the gate is opened and return is made to the caller. Input buffers may contain a pre-generated RPL which is modified when executing the RECEIVE macro. If no RPL is available, the pre-generated RPL of the SSCB is moved into the buffer and modified when executing the RECEIVE macro.

The initial return code is checked and if no error is indicated return is made to the caller. If the initial completion fails, the input buffer is chained to the PNET driver buffer queue for further processing, the gate is unlocked, and return is made to the caller.

The general flow is shown in Figure 81 on page 271.



RECEIVER TASK

** A RECEIVER task has been attached based on Request of Function Initiation. (RIF).
** Assume, the PNET driver has queued an INPUT buffer for Processing.
1. Get buffer from Received Input queue.
2. Perform deblocking of records and pass them to data-management.
3. Invoke buffer management if buffer is empty.
4. Repeat step 1-3 as long as buffers are available, Else:
** WAIT ON BUFFER **

BUFFER MANAGEMENT (IPW$$BS)

1. Dequeue buffer from RECV. Queue and provide it to receiver.
2. Queue buffer to Free-Input queue.
3. Post PNET driver ECB.

PNET DRIVER (IPW$$LD)

1. IF RECEIVE-Request indicated in NCB, take buffer out of Free-Input queue and queue it to Receive-Ahead Queue.
2. Invoke RECEIVE-manager via IPW$IOM.
3. Dequeue Buffer from LDR Buffer Queue.
4. Check Return- and Feedback Code.
5. Process Response, if applicable.
6. Process Buffer.

RECEIVE-MANAGER (IPW$$SR)

1. Prevent interlocked RECEIVE to be executed.
2. Dequeue 1st buffer from INPUT-AHEAD queue and make it to current.
3. Set RECEIVE-REQUEST if Input AHEAD queue is empty and return to caller.
4. Prepare RPL and buffer fields for receive.
5. Execute RECEIVE MACRO
6. Check Return Code: If any error, remove buffer from current and queue it to PNET driver buffer queue.
7. Return to caller.

VTAM-API

1. INITIAL Completion
2. INITIAL Completion

RECEIVE-RPL-EXIT (IPW$$SR)

1. Free RPL from VTAM, issue CHECK-MACRO
2. Take buffer out from Current and queue it to the PNET driver Buffer-Queue.
3. Reset INTERLOCK indicato.
4. Check return code and Session Termination type: If any error or session is terminating, skip to
5. If Input AHEAD Queue is not empty, call RECEIVE-Manager, else indicate RECEIVE-Request in NCB.
6. Post PNET driver ECB.
7. Return to ACF/VTAM.

VTAM-API

CHECK

BUFFER MGMNT (IPW$$BS)

1. Queue buffer to RECEIVE Queue of Receiver task.

Figure 81. PNET RECEIVE Function

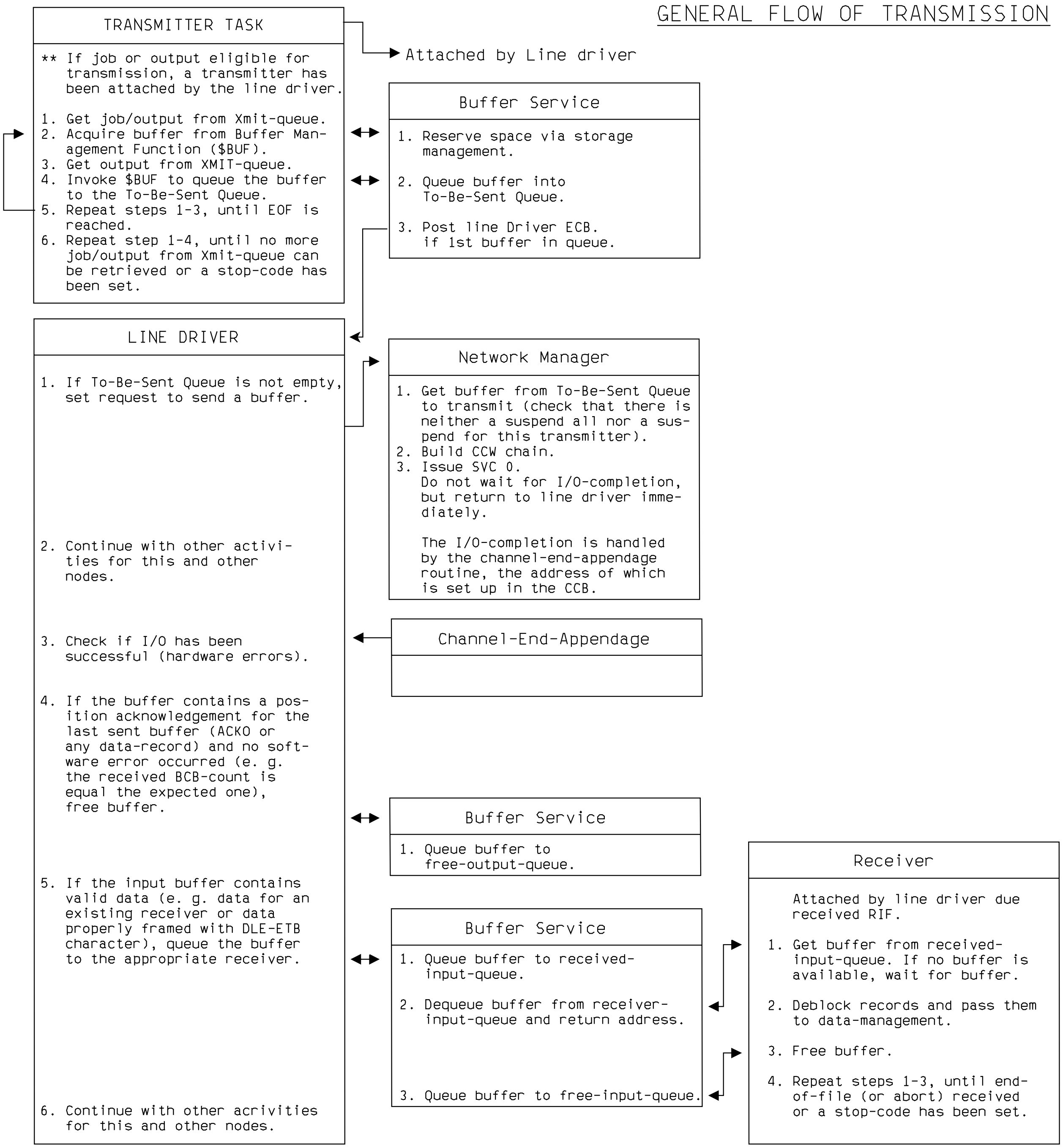
SEND Exit: The exit is scheduled by VTAM at final completion of the SEND request. The RPL is freed from VTAM use via the CHECK macro. The output buffer is then chained to the PNET driver buffer queue for further processing. If any error was detected during the CHECK, or if a session error was indicated in the NCB, immediate return to VTAM is made without initiating another SEND. If no error was detected, the SEND function is called by means of the IPW$IOM macro instruction to send another output buffer. The gate remains locked until no more buffers can be sent or an error was detected. In all cases, the PNET driver task is posted to perform the required actions for the queued buffer(s).

RECEIVE Exit: The exit is scheduled by VTAM when the actual RECEIVE completes finally. The RPL is freed from VTAM use via the CHECK macro. The current buffer is than chained to the PNET driver buffer queue for processing. If any error was detected during the CHECK or if a session error was indicated in the NCB, immediate return to VTAM is made without initiating a new RECEIVE. If no error was detected, the 'FREE-INPUT AHEAD QUEUE' is examined for a free buffer and if one is available the RECEIVE-function is called, from the exit, to start another RECEIVE. The gate remains locked until any error is detected or until no input buffer is available. In all cases, the PNET driver task is posted to perform the required actions for the queued buffer.

PNET Transmitter

The PNET transmitter (IPW$$NT) consists of two different transmitters, the job or output transmitter and the console transmitter.

Job/Output Transmitter: The PNET transmitter is responsible for transmitting jobs or output to other nodes in the network. For each active transmitter there must be an active receiver on the other end of the line. See Figure 82.



GENERAL FLOW OF TRANSMISSION

TRANSMITTER TASK

** If job or output eligible for transmission, a transmitter has been attached by the line driver.

1. Get job/output from Xmit-queue.
2. Acquire buffer from Buffer Management Function ($BUF).
3. Get output from XMIT-queue.
4. Invoke $BUF to queue the buffer to the To-Be-Sent Queue.
5. Repeat steps 1-3, until EOF is reached.
6. Repeat step 1-4, until no more job/output from Xmit-queue can be retrieved or a stop-code has been set.

Attached by Line driver

Buffer Service

1. Reserve space via storage management.
2. Queue buffer into To-Be-Sent Queue.
3. Post line Driver ECB. if 1st buffer in queue.

LINE DRIVER

1. If To-Be-Sent Queue is not empty, set request to send a buffer.
2. Continue with other activities for this and other nodes.
3. Check if I/O has been successful (hardware errors).
4. If the buffer contains a position acknowledgement for the last sent buffer (ACK0 or any data-record) and no software error occurred (e. g. the received BCB-count is equal the expected one), free buffer.
5. If the input buffer contains valid data (e. g. data for an existing receiver or data properly framed with DLE-ETB character), queue the buffer to the appropriate receiver.
6. Continue with other acrivities for this and other nodes.

Network Manager

1. Get buffer from To-Be-Sent Queue to transmit (check that there is neither a suspend all nor a suspend for this transmitter).
2. Build CCW chain.
3. Issue SVC 0. Do not wait for I/O-completion, but return to line driver immediately.

   The I/O-completion is handled by the channel-end-appendage routine, the address of which is set up in the CCB.

Channel-End-Appendage

Buffer Service

1. Queue buffer to free-output-queue.

Buffer Service

1. Queue buffer to received-input-queue.
2. Dequeue buffer from receiver-input-queue and return address.
3. Queue buffer to free-input-queue.

Receiver

Attached by line driver due received RIF.

1. Get buffer from received-input-queue. If no buffer is available, wait for buffer.
2. Deblock records and pass them to data-management.
3. Free buffer.
4. Repeat steps 1-3, until end-of-file (or abort) received or a stop-code has been set.

Figure 82. General Transmission Flow - Shown for BSC Link

Each transmitter is associated with a node. If other non-adjacent nodes are reachable via the node to which a direct connection exists, as specified via the ROUTE1 and ROUTE2 parameter of the appropriate PNODE macro, the transmitter is said to be eligible for these nodes, too. Each node currently active is described by a Node Control Block (NCB). Activation of a transmitter task is initiated by requesting task creation in the appropriate task entry, contained in the NCB, and posting the PNET driver.

A transmitter task is attached by the PNET driver in the following cases:

1. Initially when the connection between the two nodes is established and sign-on processing completed, then one job and one output transmitter are attached.
2. When a job or output eligible for transmission is put into the XMT queue and the final destination is reachable via this connected node.
3. When a PACT command is issued for the transmitter.
4. When a job or output is returned to the XMT-queue with its original disposition by means of the IPW$DQS HOLD macro. The transmitter going to be drained checks if there are other inactive transmitters of the same type and if so attaches one.
5. When a connection between two nodes is broken, then the first inactive transmitter for an alternate route node is attached.

Before attaching the transmitter the PNET driver ensures that the connection is not drained (either done by PSTOP PNET,nodeid,EOJ or PEND command given by the central operator) and equips the transmitter with a work area.

When the job or output transmitter is initially entered, the work area previously acquired by the PNET driver is initialized. The IPW$GQS macro is then executed to obtain the next eligible queue entry (job or output, dependent on the type of transmitter). The 'Get Next Queue Set' routine scans the XMT queue for a job or output which the transmitter can send and selects the oldest highest-priority job or output queue entry which is destined to, or routed via, the node which the transmitter is serving.

A particular transmitter may be eligible to send to only a few of the nodes in the network. Eligibility is determined by the 'Get Next Queue Set' routine, which determines the best path to reach any node.

If the 'Get Next Queue Set' routine returns to the transmitter without selecting a queue entry, the transmitter frees the storage area used as work space, informs the PNET driver about the termination and detaches itself. If however, a queue entry was selected, the transmitter initializes the account record with the transmission start time, date and information extracted from the queue record. If the previous transmission of a job or output was aborted, the IPW$STM macro is issued to allow the receiver on the other end of the line to do its cleanup processing before the next request for transmission (RIF) is sent.

The get data record routine is invoked via macro IPW$GDR to obtain the first data record (the job header record). The composer is called to generate a request for permission to transmit (RIF) and to schedule the buffer for transmission.

If permission is not granted (the PNET driver has set the immediate stop code in the TCB on receipt of the negative permission) the transmitter informs the central operator via message 1RA9I and the IPW$DQS HOLD macro is issued to return the queue entry to the XMT queue with its original disposition.

The transmitter task entry table, contained in the associated NCB, is scanned for any inactive transmitter of the same type. If found, the PNET driver is posted to attach that transmitter in order to attempt to transmit the just re-queued queue entry. All acquired storage areas are released and the transmitter is detached.

If permission is granted (the receiver on the other node is prepared to accept the transmission), the job header record is completed with information from the queue record (class, target destination, user-id, system-id, time event scheduling information, disposition for local processing, etc.) and the composer is called to transmit the record. After sending the job header record, the transmitter issues the IPW$GDR macro to get the records from the data file and to pass them to the composer one at a time.

When the transmitter encounters a data set header record, the record is updated with information extracted from the queue record (class, target destination, user-id, no. of copies, disposition for local processing, etc.) and the composer is called to prepare the record for transmission.

After the job trailer has been passed to the composer to be scheduled for transmission, the composer is again called to send an end-of-file indicator to the receiver. The composer then waits for acknowledgement of the end-of-file. If positive acknowledgement is received, the responsibility for the transmitted job or output has been accepted by the receiver and the transmitter removes the queue entry from the XMT queue by issuing the IPW$DQS macro, or keeps the queue entry in the queue according to the disposition.

The central operator and, if Notify was requested, the originator are informed via message 1RA0I that the transmission successfully completed. The account record is completed with the transmission stop time and the IPW$PAR macro is issued to write the account record. The queue record area and the logical data area are released and the transmitter is prepared to start with the next queue entry.

Normal Termination: A transmitter task remains active as long as there are entries in the XMT queue which are eligible to be transmitted by this task. If there are no further eligible entries in the XMT queue, the task is detached and will be attached again under above circumstances.

Abnormal Termination: If at any time the transmitter encounters an error code, this routine is entered and the transmission is aborted.

Error codes are set after the operator has entered a PSTOP, PDRAIN, or PFLUSH command, or in the case that the communication line has broken, or that the connected node has indicated that it does not have a corresponding receiver available to receive data from this transmitter. An error code is also set when an I/O error occurred while accessing the spool files and in this case the task terminator routine (IPW$$TR) branches directly to this routine after cleaning up the queue and data files.

On entry to the routine the current stop state is saved as only this stop state is required for the various abnormal termination activities. The 'to-be-sent' queue is cleared and the composer is called to send an 'ABORT' to the receiver. No 'ABORT' is sent in case of receipt of an NPGR, in case of a broken line (irrecoverable I/O error, SIGNOFF received), in case that the task termination routine was invoked prior to sending a RIF and in case that the stop state occurred after having sent the EOF signal to the receiving node.
The composer on behalf of the transmitter waits then for an acknowledgement (receiver cancel from the other side).

The cancel code is stored in the account record. The operator is informed via message 1RA9I, and in the case that the transmitter will also be drained, via message 1RA8I. The queue entry is returned to the XMT queue with a disposition requested by the saved stop state. If the transmitter is to be drained, then the PNET driver is requested to start another not drained transmitter. If applicable, additional information is written into the account record. The logical data area and queue record area are then released.

If the saved stop state does not require an immediate and unconditional termination of the transmitter task, any subsequent transmission will start again from the beginning.

Console Transmitter: The console transmitter associated with a node is responsible for sending messages and commands, in the NMR format, to that node. The task is attached by the PNET driver when a message/command is put in the message/command queue of the node concerned.

When the console transmitter is initially entered, the routine initializes the work area which was acquired by the PNET driver. The IPW$ICS REQ=GET macro is then issued to obtain the first message/command from the queue. The NMR is then passed to the composer which converts the record into the form required for transmission, including compression and blocking. After passing the record, the storage area occupied by the NMR is released by means of the IPW$ICS REQ=DEL macro instruction.

This process continues until all messages/commands are scheduled for transmission. When the message/command queue is empty, the composer is called to schedule the partially filled buffer for transmission. The composer returns to the transmitter when the last buffer has been transmitted. If, meanwhile, another message/command was put into the queue associated with the node, the console transmitter starts the whole process again. Otherwise the routine informs the PNET driver about the termination and, after releasing all prior acquired work area, detaches itself.

PNET Composer

The composer (IPW$$NC) runs under the control of the calling task and acts as a function module. It is responsible:

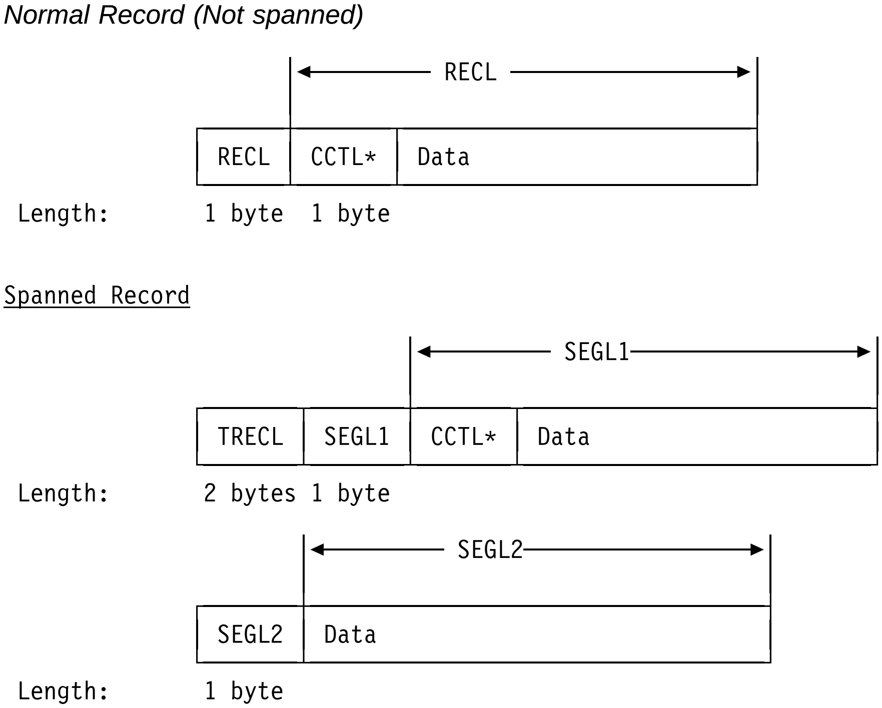
- To set up an MLI control record (eg. RIF, PGR) and to schedule the record for transmission.
- To prepare all records (e. g. data records, nodal message records, job header records etc.), with the exception of topology records, in the format required for transmission. Framing the records with RCB and SRCB for BSC/CTC or RID for SNA respectively, compressing the record and blocking.
- To queue the current buffer, if any, to the 'to-be-sent' queue.

The type of request is passed to the composer as a parameter list.

Normal Record and Nodal Message Record Processing: The routine expands the record with trailing blanks, if necessary, and compresses the records one at a time and puts the result in the output buffer. If the record does not fit into the output buffer, the buffer is scheduled for transmission by issuing the IPW$BUF MODE=OUT, TYPE=QUEUE macro instruction. After the buffer is queued, a new output buffer is obtained, if possible, by issuing the IPW$BUF MODE=OUT,TYPE=GET macro instruction. If a buffer is available, record compression continues. If, however, no buffer is currently in the free output queue and no buffer can be obtained from the VSE/POWER storage pool, the routine waits until an output buffer is returned to the free output queue by the PNET driver.

Normal data records (not segmented) are *not* spanned across buffers.

Data records are broken up into segments of a maximum length of 256 bytes. Segments belonging to one data record may reside in different output buffers, but segments always reside completely in one output buffer, i. e. segments do not span buffers. The format of output records is shown in Figure 83 on page 277.



RECL Length of data in unspanned record

CCTL Command code

TRECL Total data length of segmented record

TRECL=SEGL1+SEGL2+.....SEGLn

SEGL Length of data in segment

* means that this field is optional.

Figure 83. Format of Output Records

Network Control Record Processing: Network control records are segmented into records with a maximum length of 256 bytes. Each segment is provided with a 4-byte header containing a sequence count and a continuation indicator. The first segment of each network control record must be the first record in the output buffer. The buffer may then contain other records after the control record.

MLI Control Record Processing: All MLI control records, with the exception of ABORT, are built in small output buffers of fixed length and of a size capable of holding any type of MLI record. The buffers are obtained by means of

```
IPW$BUF MODE=OUT,TYPE=CNTRL
```

In the case of a request for an ABORT, a check is made to see whether an output buffer still exists which is not yet queued to the 'to-be-sent' queue. If so the ABORT is indicated in the first RCB/SRCB of this buffer, otherwise a control record is used.

The data transmission buffer status, after it has been written, is set into the output buffer header. The PNET driver reads this status and, after the buffer has been successfully transmitted, stores it into the task TCB for later action.

PGR, NPGR, Transmission Complete, ABORT, and Receiver Cancel, do not require a response from the receiving system. To ensure proper synchronization between the sending and receiving nodes, the BUFPOST bit is set in the output buffer, which causes the PNET driver to post the composer after transmitting one of these records and a wait is established. Since a RIF and EOF synchronize with PGR/NPGR respectively transmission complete, a wait must also be issued for these MLI records.

The composer is re-activated by the PNET driver when:

- A PGR, NPGR, Transmitter Cancel, Receiver Cancel, or ABORT has been received from another node.
- Sending a PGR, NPGR, Transmitter Cancel, or Receiver Cancel, and the MLI buffer has been transmitted.
- The local operator has entered a PSTOP or PDRAIN command, or a SIGNOFF has been received, or an unrecoverable I/O error has occurred, or a transmission sequence error has occurred.

Note: MLI control records are not compressed in BSC mode. In SNA mode, each MLI record is regarded as a complete request unit (RU) and as such is compressed.

PNET Receiver

The PNET receiver (IPW$$NR and IPW$$NR2) consists of two different receivers, the job or output receiver and the console receiver. The code for both kinds of transmitters is in the two modules IPW$$NR and IPW$$NR2. Each of the two modules contains part of the following function, separated just because the code was too much for one module.

Job/Output Receiver: The receiver is responsible for receiving either jobs or output transmitted from another node, and writing them to the VSE/POWER spool files according to their queue identifiers. VSE/POWER supports up to eight receivers concurrently active. The eight receivers may be a mixture of job and output receivers, however the maximum number of one type is 7.

For each active receiver there must be an active transmitter on the other end of the line.

The receiver calls the presentation service routine, which is responsible for obtaining physical buffers from the received input queue pertaining to the receiver, de-blocking and de-compressing each record and passing the logical record, on demand, to the receiver.

The receiver is attached by the PNET driver when the corresponding transmitter on the other end of the line requests permission to transmit by sending a 'request to initiate a function' (RIF) control record. If, however, either VSE/POWER is in shutdown period or the node is to be terminated, the PNET driver rejects that request by sending a negative permission (NPGR) control record.

When the receiver is initially entered, the work area already acquired by the PNET driver is initialized. The receiver then checks if any termination code has been set in the TCB (for example the operator has drained the receiver) and if so, the receiver terminates by calling the composer in order to send a negative permission to the corresponding transmitter. Otherwise the composer is called to prepare a 'permission granted' (PGR) control record and to schedule the record for transmission. After the buffer has been successfully sent, the composer returns to the receiver which then calls presentation service in order to obtain the first logical record.

If presentation service returns with a bad return code, the receiver aborts the current data stream. Otherwise the record control byte (RCB) and subrecord control byte (SRCB) which are returned by presentation service, are checked to determine what type of record was received (job header, job trailer, data set header, end-of-file, abort, normal data record, or nodal message record). The layout of the RCB can be found in Figure 88 on page 287 and that of the SRCB in Figure 89 on page 288.

The received record type is checked against the record type associated with the receiver; only records pertaining to the receiver and nodal message records are expected. If the received record was not expected, the receiver aborts the transmission.

If a network user exit is present, any network control records (job header, job trailer and data set header) as well as any JECL and JCL statement are passed to the exit before the appropriate routine is called which handles the type of record that was received. Before entry to the user exit a default switch to non-parallel (NP) mode is done to allow for Supervisor Control Block update by the exit. If however the loading conditions of the exit specify 'PA', this extra switch is suppressed. Upon return from the user exit an unconditional switch to parallel mode is done. For details refer to 'Multiprocessor Support'. If, however, the user exit indicates that deletion of the record is required, then the receiver branches to obtain the next record from presentation service.

Note: Network control records may not be deleted by the user exit.

Job Header Record Processing: A job header record is expected only once per job and must be the first record received. If such a record is received in the middle of a transmission, the receiver aborts the transmission.

When a job header record is received, the receiver prepares to receive the job/output.

- The IPW$RQS macro is issued to reserve a queue record and data file space.
- Storage for a data block is obtained by issuing the IPW$RSV macro instruction.
- The queue record is constructed from the information in the job header record (general section and if present VSE/POWER section).
- Work area is reserved to hold a copy of the job header record in storage. This is necessary because at any point within the transmission a new queue entry might be created and the first record that must be spooled has to be the job header record.

When output is being received, both the queue record and the data block are anchored to a new control block, referred to as the data set control block (DSCB). The DSCB contains, besides the anchors, information about the characteristics of the output data set (forms, class, priority, FCB, etc.).

If all output data sets for a job have the same routing and all require the same forms, FCB, UCB train, etc., and all have the same class and priority and the data stream sections, if present, are identical then only one queue entry is created to represent all of the data sets. If any of the output data sets differ from one another in class, destination, type, etc, then multiple queue entries are created.

A new VSE/POWER job number, which is used only during the time the job/output is resident on this node, is then assigned to the queue record. The original job number which was assigned to the job when it entered the network, remains unchanged in the job header record. If the received data set is a "spin off" data set or a segmented output queue entry, the original job number, obtained from the job header record is used to facilitate easier queue manipulation.

The account record, located in the receiver work area, is initialized with the transmission start time, date and information extracted from the queue record.

The priority obtained from the job header record is converted into the VSE/POWER priority range according to following table. See Figure 84.

NJE Priority	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
VSE/POWER	1	1	2	2	3	3	4	4	5	5	6	7	7	8	8	9

Figure 84. JES2 to VSE/POWER Priority Conversion Table

If any on the initialization functions fails, the receiver branches to abort the transmission. When the receiver has completed initialization for the job/output, the job header record is written to spool. Next the GET record main loop is entered to get the next record from presentation service.

Data Set Header Record Processing: When a data set header record is received, the receiver performs the initialization required before receiving the job or output. When receiving a job, the data set header record consists only of a record characteristics change section, defining a different record length and/or format (other than 80). The record length is copied from the data set header record to the queue record. If a VSE/POWER section is present, the 3540 cuu address is also copied into the queue record and the TCB is flagged that now 3540 data records follow.

The JES2 output transmitter may send more than one data set header record before sending the data records. In this case, each data set header record may indicate a different destination for the same data set. The receiver splits each data set into multiple queue entries (data sets) - one for each destination. This means that when the data stream is transmitted again, it is sent as multiple streams, each containing the data set and each independent of the other.

For every data set header record so received, a data set control block (DSCB) is built, including queue record allocation and reserving of data block space. The DSCB is placed in the active DSCB chain. Each subsequent data record is now spooled for each queue entry in the active DSCB chain. The DSCB contains all required spooling information for the queue and data file.

If another data set header record is received after data has been received, then the receiver suspends the spooling of all queue entries currently in the active DSCB chain by writing the current DBLK with the end of block indication and releasing the data block buffer afterwards. All data sets in the active DSCB chain are then placed in the suspended DSCB chain.

The suspended DSCB chain is then scanned for a match with the new data set header record. A match is found when the output characteristics such as forms id, FCB, destination, etc., the output class and priority are identical.

If such a DSCB is found, which means that there already exists an incomplete queue entry which can be further used, the DSCB is removed from the suspended DSCB chain and put into the active DSCB chain. Additionally, buffer space for the data block is reserved and anchored to the DSCB.

If no DSCB is found, a new DSCB is created and initialized with the information from the job header record and the just received data set header record. A new queue record is reserved by means of the IPW$RQS macro and storage for the data block is obtained. Both the queue record and the data block are anchored to the DSCB being built. The job header record and the data set header record are then written as first records on spool.

Data Record Processing: When receiving a job, the record is written on spool by issuing the IPW$PDR macro. The account information is updated accordingly. The record is spooled for each queue entry described by a DSCB entry in the active DSCB chain. The account record is updated accordingly.

Nodal Message Record Processing: The record is passed to the message distributor for further processing by issuing the IPW$GMS TYPE=DIST macro instruction.

Job Trailer Record Processing: After all data has been transmitted, the transmitter sends the job trailer record. When the receiver receives the job trailer record it starts termination processing of the job or output. The job trailer is written, with the end of data indicator set, to spool. Any DSCB entry currently in the suspended DSCB chain is placed in the active DSCB chain. After the job trailer is written, the data block is released and returned to the VSE/POWER storage pool. The receiver turns off the job boundary flag to indicate that the next record should be end-of-file.

End-of-File Record Processing: When end-of-file is received, each queue entry, described by a DSCB in the active DSCB chain, is added to the queue according to its class and priority by means of the IPW$AQS macro instruction. If, however, the end-of-file record was not preceded by a job trailer record, the receiver discards any queue entry by branching to abort the transmission.

When a job or output queue entry is added to the class chain it appears as if it had been read in, or completed execution, on the receivers system.

Next the receiver checks to see if any of the received data sets are destined for printing or punching on this node. If so, and the job header contains a NOTIFY user/remote id, the receiver issues message 1RB5I via the IPW$NTY macro to inform the user or the remote id that output has arrived. The queue record area and the data block buffer are released and returned to the storage pool. This process is repeated for each queue entry in the active DSCB chain. Finally the account record is completed with the transmission stop time and then it is written by issuing the IPW$PAR macro instruction.

Normal Termination Processing: The receiver calls the composer in order to send a positive acknowledgement. The composer returns to the receiver after the record has been successfully sent. The receiver then releases the storage occupied by the job header record, alternate presentation buffer area and user record area, if any still exists. Finally the PNET driver is informed that the receiver is going to be detached.

Abnormal Termination: If at any time the receiver encounters an error code as posted by the PNET driver indicating that the communication line has been broken or that the transmitter sent an abort record to abort its job or output transmission this routine is entered.

The transmission is also aborted if an error is detected in any of the received job header, job trailer or data set header records, if records are received out of sequence (an unexpected record received), if the operator terminated the transmission by means of the PSTOP, PDRAIN or PFLUSH command, or if an I/O error occurred while writing either a data block or queue record to spool. In the latter case, the task termination routine (IPW$$TR) branches immediately to this routine after cleaning up the queue and data file. In general the task terminator routine has already removed the bad queue entry causing the I/O error from the class chain.

The operator is informed about the unsuccessful transmission via message 1RB6I. Any DSCB currently in the suspended DSCB chain is put in the active chain. Each spool space, allocated by the incomplete queue entry is then freed via the IPW$DQS and IPW$FQS macro instructions and its associated storage areas are released. Depending on the actual status the transmission is aborted by calling the composer to send either a negative permission or a receiver cancel record.

When a receiver cancel record is sent, the receiver enters a data stream loop, waiting for an abort or end-of-file record from the transmitter. Each record received while in this condition is purged, unless it is a nodal message record which is passed to the message distributor via the IPW$GMS TYPE=DIST macro instruction. The purge loop is left when either the PNET driver encounters a line error or a sign-off record is sent or received. In this case the PNET driver propagates the error condition to the receiver. Any work space previously acquired, such as the job header record area or the alternate presentation buffer area, are released and returned to the VSE/POWER storage pool.

If applicable, message 1RA8I is issued to inform the operator that the receiver is now drained. Any buffer still in the receiver input buffer queue is released and its storage returned to the VSE/POWER storage pool. Finally the account record is completed with the transmission stop time and the account record is written by means of the IPW$PAR macro instruction. The cancel code is set to X'50'. The PNET driver is informed that the receiver is going to be detached.

JECL Scanner at Receive Time: If a job for the local reader queue is received, the received records are scanned for * $$ JOB statements. The syntax checking is done by calling IPW$$SC using the interface macro IPW$SRJ. The consistency checking for time event scheduling operands is done by calling IPW$$LR using the entry address CALR within the CAT. The parameters (input and output) are passed to IPW$$LR using the mapping macro IPW$DLW. If an error has been found, IPW$$SC does not issue any error message. IPW$$NR2 issues the appropriate error message to the local operator as well as to the

originator or to the user at the node defined by the notify parameters. If an error has been found during the consistency check, IPW$$LR issues the error message to the local operator and IPW$$NR2 sends the error message to the originator or to the user at the node defined by the notify parameters.
In order to minimize the number of extra fields in work areas and support the ability to receive more than one VSE/POWER job in one network job, the DSCBs, so far used only for the output receivers, are used for a job receiver as well. Thus the following fields are used for the job receiver, sometimes in a little different way than for an output receiver:

1. NRDSSUSP contains the address of a DSCB in which pointers are saved for
   a. the job header record of the network job in DSOPTBAD
   b. the queue record initiated with values out of the job header record of the network job in DSTCBQV.

   These fields are only used, if an * $$ JOB statement has been found.
2. NRDSACT contains the address of a DSCB which is the first element of a DSCB chain, each DSCB element corresponds to one VSE/POWER job within the received network job.
   This field is only used, if several VSE/POWER jobs have to be spooled.
3. NRDSINIT is used as before and contains the address of a DSCB which belongs to the VSE/POWER job just being received. The fields in the DSCB are set at the time the DSCB is chained to the active DSCB chain, i.e. another VSE/POWER job has to be created.

In order to waste not too much storage, the storage for the DBLK area is reused, if more than one VSE/POWER job is received within one network job.

The first record to be received is a job header record. If the job is a reader job and destined for this node, the job header record is not yet written. As soon as the next record is received, the type of the record is checked. If it is no * $$ JOB statement, the job header record is written and thereafter the just received data record. If it is an * $$ JOB statement, the * $$ JOB statement is processed and a new job header record is built using the specified parameters of the * $$ JOB statement or the VSE/POWER defaults. If necessary any continuation statements are processed. When processing an * $$ JOB statement, this statement is translated to uppercase. Even the old positional format of an * $$ JOB statement is accepted, although it is no longer described in the documentation. As soon as the end of an * $$ JOB statement has been detected the job header record is written to the spool file. If once an * $$ JOB statement has been found and now another * $$ JOB statement is received (i.e. without having received an * $$ EOJ statement before), the so far received job is written to the spool file and a new job is created.

When receiving a job trailer record:

1. If NRDSACT (active DSCB entry) is zero, i.e. just one VSE/POWER job has been received, the job trailer record is written to the spool file.
2. If NRDSACT (active DSCB entry) is not zero, i.e. more than one VSE/POWER job has been received, the DSCB pointed to by NRDSINIT is added to the active DSCB chain and the job trailer record is written to the spool file.
3. Note: if more than one VSE/POWER job is received within one network job, each time a new VSE/POWER job is created, the job trailer record is written for the previous VSE/POWER job.

When receiving the end of file record, the job is added to the spool file:

1. If NRDSACT (active DSCB entry) is zero, i.e. just one VSE/POWER job has been received, this job is added to the spool file right now.
2. If for a job receiver a suspended DSCB exists, release storage for saved job header record of the network job, queue record and DSCB entry, if necessary (once an * $$ JOB statement has been found). The field NRDSSUSP has been used to save the DSCB address. The field DSOPTBAD has been used to save the address of the job header record of the network job and add all active entries to the spool file.

PNET Presentation Service

The IPW$$NP routine is responsible for obtaining physical buffers from the 'received input queue' and passing decompressed logical records one at a time to the receiver. The presentation service routine runs as a function module under the TCB of the calling task.

The logical record is decompressed in the presentation buffer appended to the presentation service work area passed by the caller. The decompression routine in the IPW$$NK module is used for the decompression of the records.

For SNA, each data record or segment in the buffer (RU) is preceded by the record identification field (RID) during compression at the transmitting node. The RID identifies the type of record and contains the length of the decompressed data record. Since the RID has undergone compression at the transmitting node, it must go through decompression at the receiving node. Thus the decompression routine is called to decompress a RID into a 3-byte work area in the buffer header. If the record is not an abort or end-of-file record, the length is obtained from the RID and the decompression routine is invoked again to decompress that many bytes from the buffer into the presentation buffer.

All records passed to the receiver have the following format:

	RCB	SRCB	Length	Data
Length:	1	1	2	n

RCB - Record control byte
SRCB - Subrecord control byte
Length - length of original record
data - actual record

Figure 85. PNET Internal Record Format

When no buffer is currently in process or the buffer has been emptied, a new buffer from the received input queue associated with the task is obtained by issuing the IPW$BUF MODE=IN,TYPE=GET macro instruction. The macro expands into a linkage to the buffer service routine which de-queues the first buffer from the received input queue and passes its address back to the caller. If the received input queue is empty, the buffer service routine waits until a buffer arrives or an error condition occurs, resulting in an immediate termination of the receiver. (The PNET driver posts the task after an input buffer for the task is received.)

Three types of record may be received:

1. Data records
2. Network control records (job header, job trailer, etc.)
3. Normal or abnormal end-of-file (data)

When a data record is received, the routine checks to see whether it is a spanned record and if no it is passed to the caller. The maximum length of a record may be 32.767 bytes which requires that the transmitter breaks the record up into pieces, no larger than 256 bytes, and transmits each piece as a single logical record. This routine re-constructs the record, from the received pieces, into the original record format.

If the presentation buffer is not large enough to hold the entire data record, an alternate presentation buffer in the length of the total record plus some control bytes is acquired. The alternate presentation buffer is released the next time the routine is entered. Each piece is decompressed in the presentation buffer and the 4 control bytes are removed. (See Figure 85 for details.) If an error is encountered, a return code is passed to the caller.

Since there is no restriction on the length of a job header, job trailer or data set header record, they may require more than one logical record (limited to
256 byes in length) for transmission. If the header/trailer record occupies more than one logical record, the presentation service routine rebuilds the entire header/trailer before passing it to the receiver. As each piece is received its sequence number is verified and if an error is encountered a return code is passed to the caller.

Additionally, the general, VSE/POWER and 3800 sections are checked for the minimum required length. If the section is too short, the routine expands it to the required length by moving down any data following the general section and padding the section with binary zeros. If the section is expanded, both the overall length and the section length are updated. This is necessary, because the section may be enlarged from one release to another and not all nodes within a network may have the same release level.

If normal or abnormal end-of-file is received, appropriate flags are set in the TCB of the calling receiver and immediate return to the caller is made.

If the buffer is processed (all logical records have been passed to the receiver), the buffer is returned to the free input queue by issuing the IPW$BUF MODE=IN,TYPE=FREE macro instruction.

PNET Buffer Service

The IPW$$BS routine is responsible for queuing all incoming buffers to the 'received input queue', to queue all outgoing buffers in the 'to-be-sent queue', and to supply buffers for both the transmitters and receivers. The buffer service routine runs under control of the calling task and acts as a function module. The routine is called by means of the IPW$BUF macro instruction.

The buffers required to process BSC or CTC nodes are provided from real storage, while those required to process SNA or TCP or SSL nodes are provided from virtual storage (GETVIS-24).

For a detail description of the functions provided, see Appendix C, “VSE/POWER Internal Macros” - “IPW$BUF - Invoke PNET Buffer Service” on page 749.

PNET Compression/Decompression

The PNET compression function (IPW$$NK) is used to condense duplicate character strings to reduce tele-processing transmission volume and thus transmission time. the string is replaced with two In the case of three or more duplicate non-blank characters, bytes: a 'String Control Byte' (SCB) followed by the character itself. Two or more duplicate blank (X'40') characters require only the SCB indicating the character string length. Strings of non-duplicate characters are also preceded by a SCB to indicate the length. The decompression function uses the SCBs to reconstruct the original data.

The SCBs and the method of compression/decompression differ according to the caller's indicated TP line discipline (BSC/CTC/TCP/SSL or SNA). For compressed BSC/CTC/TCP/SSL data only, each record ends with a special end-of-record SCB. Therefore BSC/CTC/TCP/SSL compression appends this SCB to each compressed data string offered as input.

BSC/CTC/TCP/SSL decompression begins with the location specified in an input parameter list, and stops when the end-of-record SCB occurs. SNA decompression begins with the location defined in a parameter list but continues until the given length (in RID) of output is obtained or all input is processed. If the SNA input buffer requires more output space than available for decompression, then the input is modified to insert an SCB at the input location where the next decompression is to take place and a pointer is passed to the caller specifying where the next decompression should begin.

SNA decompression has a further function enabling the caller to 'sneek-a-peek', i.e. the caller may specify a small output buffer (e.g. 3 bytes), and the decompression routine will fill as much of the output buffer as possible before closing and returning to the caller without modification of the input. This enables the PNET driver to decompress the received record identifier (RID) in order to determine the stream control action for the node being processed before the data is actually decompressed. Likewise the PNET SNA presentation service can determine the length of the record following the RID, (the length is in the second byte of the SRCB), that should be decompressed to obtain the complete record.

The SNA decompression routine also performs decompaction, if the input data stream contains compacted data and a Compaction Table Block (CMPT) is available. A Compaction Table Block is only built at session establishment, if VSE/POWER receives a valid Function Management Header 3 (FMH3), containing the compactable characters (master and non-master characters).

The SCBs are described in Figure 86.

SCB Byte Codes:		SCB Function:
SNA:	BSC:	
N/A	0000-0000	End-of-record indicator.
00cc-cccc	11cc-cccc	Non duplicate character string. 'cccccc' is the number of characters in the string. The string of characters immediately follows this SCB.
01ee-eeee	N/A	Compacted character string. 'eeeeee' is the number of characters in the string to be decompacted.
11dd-dddd	101d-dddd	Duplicate character string. The character following the SCB was duplicated 'ddddd' times.
10bb-bbbb	100b-bbbb	Blank string. 'bbbbb' is the number of blank characters in the string.
01aa-aaaa	------	Compacted character string. 'aaaaaa' is the number of compacted bytes following the SCB.

Figure 86. SCB Byte Codes

The input parameter list that is required when calling the routine as well as the possible error return codes, are described in the layout of the compression work area (see “Network Compression Work Area” on page 552).

PNET Multi-Leaving Format

The basic element for multi-leaving transmission is the character string. One or more character strings are formed from the smallest external element of transmission - the logical record. For efficiency in transmission, each record is reduced to a series of character strings of two basic types:

- A variable length non-identical series of characters
- A variable number of identical characters

Because of the frequent occurrence of blank characters, a special case is made for identical characters when the duplication character is a blank. A 1-byte control field, called a string control byte (SCB), precedes each character string to identify the type and length of the string. Thus a string of non-identical characters is represented by an SCB followed by the non-duplicate characters. A string of consecutive, duplicate, non-blank characters is represented by an SCB and a single character. The SCB indicates the duplication count and the character following indicates the character to be duplicated. In the case of an all-blank character string, only an SCB is required to indicate both the type and number of blank characters. Figure 87 describes the supported SCBs.

Binary	Meaning
0000 0000	End-of-record If first SCB, this also indicates end-of-file.
0100 0000	Abort transmission
100b bbbb	"bbbbb" blanks are to be inserted.
101d dddd	The single character following this SCB is to be duplicated "ddddd" times.
11cc cccc	The "cccccc" characters following this SCB are to be inserted.

Figure 87. String Control Byte (SCB) for BSC/CTC/TCP/SSL communication

A data record to be transmitted is, therefore, segmented by the transmitting program into the optimum number of characters to take full advantage of the identical character compression. A special SCB is utilized to indicate the grouping of character strings which compose the original logical record. The receiving program can then re-construct the original record for processing.

In order to group logical records together in a single transmission block, an additional 1-byte control field precedes the group of character strings representing the original logical record. This field, the record control byte (RCB), identifies the general type and function of the logical record (input stream, print stream). A particular RCB type has been designated to pass control information between the various systems. To provide for simultaneous transmission of similar functions (such as multiple input streams), a stream identification code is included in the RCB. Figure 88 on page 287 shows the various supported RCBs.

Binary	Hex	Meaning
0000 0000	00	End-of-Block
riii tttt	01-7F	Reserved for future use
100 0000	80	Reserved
1001 0000	90	Request to initiate function (RIF) (SRCB=RCB of function)
1010 0000	A0	Permission to initiate function (PGR) (SRCB=RCB of function)
1011 0000	B0	Negative permission or receiver cancel (NPGR) (SRCB=RCB of function)
1100 0000	C0	Acknowledge transmission complete (ACT) (SRCB=RCB of function)
1101 0000	D0	Inform receiver initiated (SRCB=RCB of stream)
1110 0000	E0	BCB sequence error
1111 0000	F0	General control record
1001 0001	91	RJE console message
1iii 0001	A1-F1	Reserved for future use
1001 0010	92	RJE operator command
1iii 0010	A2-F2	Reserved for future use
1iii 0011	93-F3	RJE input record
1iii 0100	94-F4	RJE print record
1iii 0101	95-F5	RJE punch record
1iii 0110	96-F6	Data set record
1iii 0111	97-F7	Terminal message routing request
1iii 1000	98-F8	NJE input record (98/A8/... job xmt/rcv)
1iii 1001	99-F9	NJE SYSOUT record (99/A9/... output xmt/rcv)
1001 1010	9A	NJE operator command/NJE console message
1iii 1010	AA-FA	Reserved for future use
1001 1011	9B	Reserved
1iii 1011	AB-FB	Reserved for future use
1iii 1100	9C-FC	Reserved for future use
1iii 1101	9D-FD	Not Used
1iii 1110	9E-FE	Not Used
1iii 1111	9F-FF	Not Used

Figure 88. Record Control Byte (RCB)

A second 1-byte field, the subrecord control byte (SRCB) is included immediately following the RCB. This field supplies additional information concerning the record to the receiving program, for example, in the transmission of data to be printed, the SRCB can carry carriage control information. Figure 89 on page 288 shows the layout of the supported SRCBs.

RCB	SRCB
00	None
90	RCB of function to be initiated
A0	RCB of function to be initiated
B0	RCB of function to be canceled
C0	RCB of function which is complete
D0	RCB of initiated receiver
E0	Expected count (received count is in BCB)

F0	An identification character as follows:
	A = Initial RJE SIGN-ON
	B = Final RJE SIGN-OFF
	C = Print initialization record
	D = Punch initialization record
	E = Input initialization record
	F = Data set transmission initialization
	G = System configuration status
	H = Diagnostic control record
	I = Initial network SIGN-ON
	J = Response to initial network SIGN-ON
	K = Reset network SIGN-ON
	L = Accept (concurrence) network SIGN-ON
	M = Add network connection
	N = Delete network connection
	O-Z = Reserved for future use

91	1000 0000 (X'80')
92	1000 0000 (X'80')
93-F3	1000 0000 (X'80')
94-F4	Carriage control information as follows: 1010 00nn - Space immediately "nn" spaces 1011 cccc - Skip immediately to channel "cccc" 1000 00nn - Space "nn" lines after print 1000 1100 - Load printer FCB image 1001 cccc - Skip to channel "cccc" after print 1000 0000 - Print and suppress space
95-F5	1000 1111 (X'8F')
96-F6	Undefined
97-F7	Undefined
98-F8	NJE input control information as follows: 1000 0000 - Normal input record 1100 0000 - Job header 1110 0000 - Data set header 1101 0000 - Job trailer 1111 0000 - Data set trailer (not used)
99-F9	NJE SYSOUT control information as follows: 10cc 0000 - Carriage control type as follows: 1000 0000 - No carriage control 1001 0000 - Machine carriage control 1010 0000 - ASA carriage control 1011 0000 - CPDS record 11cc 0000 - Control record as follows: 1100 0000 - Job header 1110 0000 - Data set header 1101 0000 - Job trailer 1111 0000 - Data set trailer (not used) 10cc ss00 - Spanned record control as follows: 1000 ..00 - No carriage control 1001 ..00 - Machine carriage control 1010 ..00 - ASA carriage control 1011 ..00 - CPDS record 10.. 0000 - Normal record (not spanned) 10.. 1000 - First segment of spanned record 10.. 0100 - Middle segment 10.. 1100 - Last segment
9A	1000 0000 (X'80')
9B	1000 0000 (X'80')

Figure 89. Subrecord Control Byte (SRCB)

For multi-leaving transmission, a variable number of records may be combined into a variable block size, as indicated previously, (that is, RCB,SRCB,SCB1,SCB2,....SCBn,RCB,SRCB,SCB1,... etc.).

The multi-leaving design provides for two or more systems to exchange transmissions blocks containing multiple data streams in an interleaved fashion. To allow optimum use of this capability, however, a system must have the capability to control the flow of a particular data stream while continuing normal transmission of all others. For example, during the simultaneous transmission of two data streams to a system for immediate transcription to I/O devices of different speeds, such as print streams. To meter the flow of individual data streams, a function control sequence (FCS) is added to each transmission block. The FCS is a sequence of bits, each of which represents a particular transmission stream. The receiver of several data streams can temporarily stop transmission of a particular stream by setting the corresponding FCS bit off in the next transmission to the sender of that stream. The stream can subsequently be resumed by setting the bit on. The layout of the FCS can be found in Figure 90.

Binary	Meaning
r... .... r...	Reserved (must be 1... .... 1... ....)
.0..	Normal state
.1..	Suspend all stream transmission
..rr .... ..rr	Reserved for future use
.... .1..	Remote console stream identifier
.... 1...	Function stream identifier for: NJE job transmission stream number 1
.... .1..	Function stream identifier for: NJE job transmission stream number 2 NJE SYSOUT transmission stream number 7
.... ..1.	Function stream identifier for: NJE job transmission stream number 3 NJE SYSOUT transmission stream number 6
....1	Function stream identifier for: NJE job transmission stream number 4 NJE SYSOUT transmission stream number 5
.... 1...	Function stream identifier for: NJE job transmission stream number 5 NJE SYSOUT transmission stream number 4
.... .1..	Function stream identifier for: NJE job transmission stream number 6 NJE SYSOUT transmission stream number 3
....1.	Function stream identifier for: NJE job transmission stream number 7 NJE SYSOUT transmission stream number 2
....1	Function stream identifier for: NJE SYSOUT transmission stream number 1

Figure 90. Function Control Sequence (FCS)

For error detection and correction purposes, a block control byte (BCB) is added as first character of each block transmitted. The BCB, in addition to control information, contains a modulo 16-block sequence count. This count is maintained and verified by both the sending and receiving systems to prevent lost or duplicated transmission blocks. The layout of the BCB can be found in Figure 91 on page 290.

Binary	Meaning
r...	Reserved (must be 1)
.xxx	Control information as follows: .000 .... - Normal block .001 .... - Bypass sequence count validation .010 cccc - Reset expected block sequence count to "cccc" .011 .... - Reserved for future use .100 .... - Reserved for future use .101 .... - Not Used .110 .... - Not Used .111 .... - Reserved for future use
.... cccc	Modulo 16-block sequence count

Figure 91. Block Control Byte (BCB)

In addition to the normal binary synchronous text control characters (STX,ETB, etc.), multi-leaving uses two of the BSC control characters, ACK0 and NAK. ACK0 is used as filler to maintain communication when data is not available for transmission. However, VSE/POWER PNET sends always an empty block instead of an ACK0. NAK is used as the only negative response and indicates that the previous transmission was not successfully received.

Figure 92 indicates the format for an SNA transmission block:

Buffer	Meaning of Field in Buffer	Length of Field
RID	SNA NJE Record Identifier	3 bytes
Data	Logical Record	Max. 256 bytes
RID	SNA NJE Record Identifier	3 bytes
Data	Logical Record	Max. 256 bytes
RID	SNA NJE Record Identifier	3 bytes
Data	Logical Record	Max. 256 bytes

Figure 92. Multi-Leaving Buffer Format for SNA Communication

Note: Record identifiers (RID) are 3-byte headers which are required on every logical record sent or received. The RID consists of RCB - 1 byte, SRCB - 1 byte, Length(uncompressed data) - 1 byte.

An RU consists of as many logical record identifiers and corresponding logical records as will fit in the specified RU size as specified by the BUFSZ parameter. No logical record may be larger than 256 bytes plus 3 bytes for the record identifier(RID). PNET sends only one type of record (network topology, stream control or data record) within a SNA RU. PNET compresses an entire RU, i.e. everything from the beginning to the end of the RU, without regard to record identifiers.

Figure 93 on page 291 indicates the format of a typical multi-leaving transmission block for BSC/CTC/TCP/SSL:

DLE	BSC Control Character, X'10'
STX	BSC Control Character, X'02'
BCB	Block Control Byte
FCS	Function Control Sequence
FCS	Function Control Sequence (Continued)
RCB	Record Control Byte for Record 1, X'00' for null buffer
SRCB	Subrecord Control Byte for Record 1
SCB	String Control Byte for Record 1
data	Character string
SCB	String Control Byte for Record 1
data	Character string
SCB	Terminating SCB for Record 1 (end-of-record), X'00'
RCB	Record Control Byte for Record 2
SRCB	Subrecord Control Byte for Record 2
SCB	String Control Byte for Record 2
SCB	Terminating SCB for last record, X'00'
RCB	Transmission Block Terminator (end-of-block), X'00'
DLE	BSC Control Character, X'10' (not used for CTC/TCP/SSL)
ETB	BSC Control Character, X'26' (not used for CTC/TCP/SSL)

Figure 93. Multi-Leaving Buffer Format for BSC/CTC/TCP/SSL

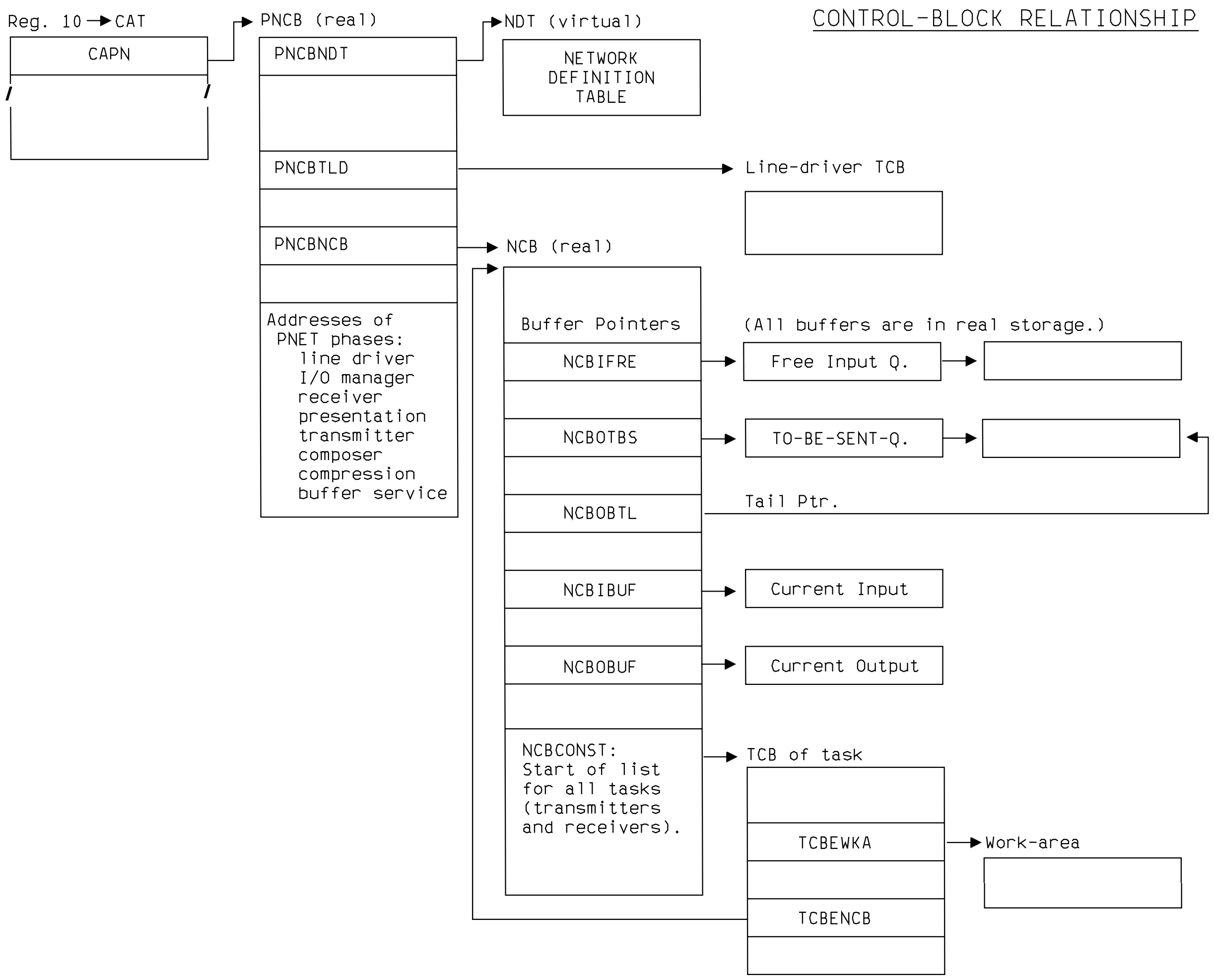


Figure 94. PNET BSC/TCP/SSL Control Block Relationship

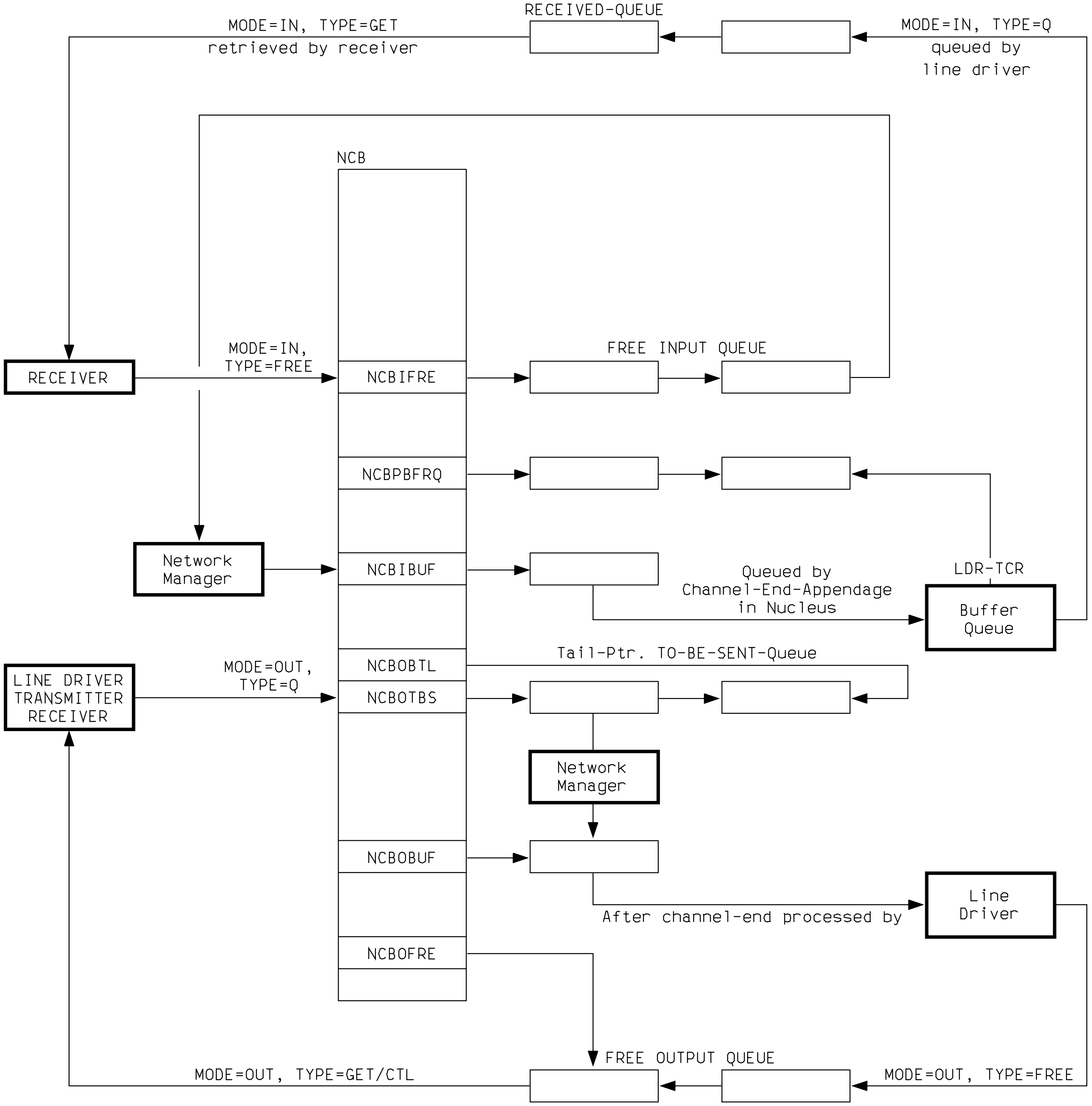
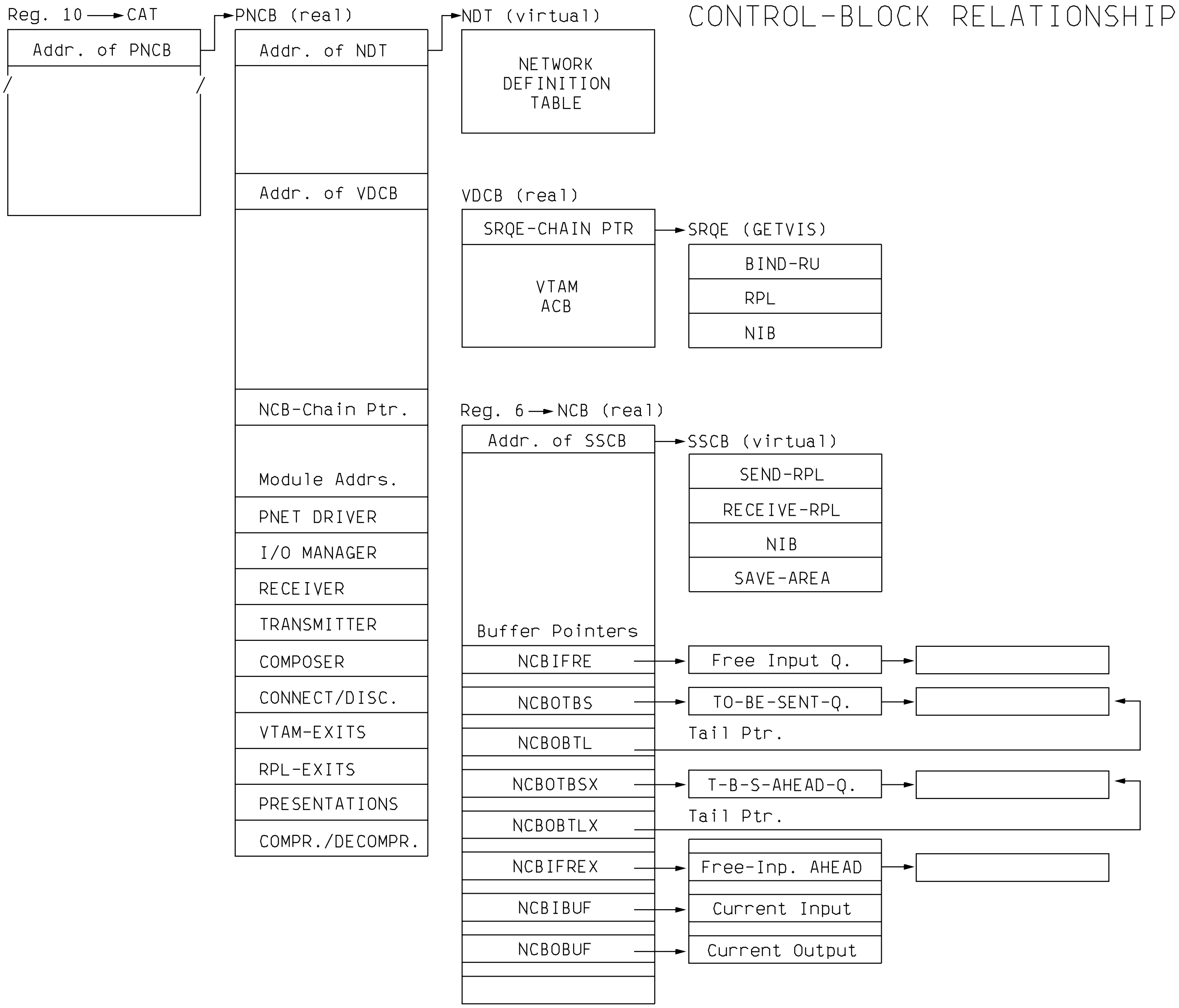


Figure 95. PNET BSC/CTC/TCP Buffer Relationship and Queuing



NOTE:
All buffers are in virtual storage.

Figure 96. PNET SNA Control Block Relationship

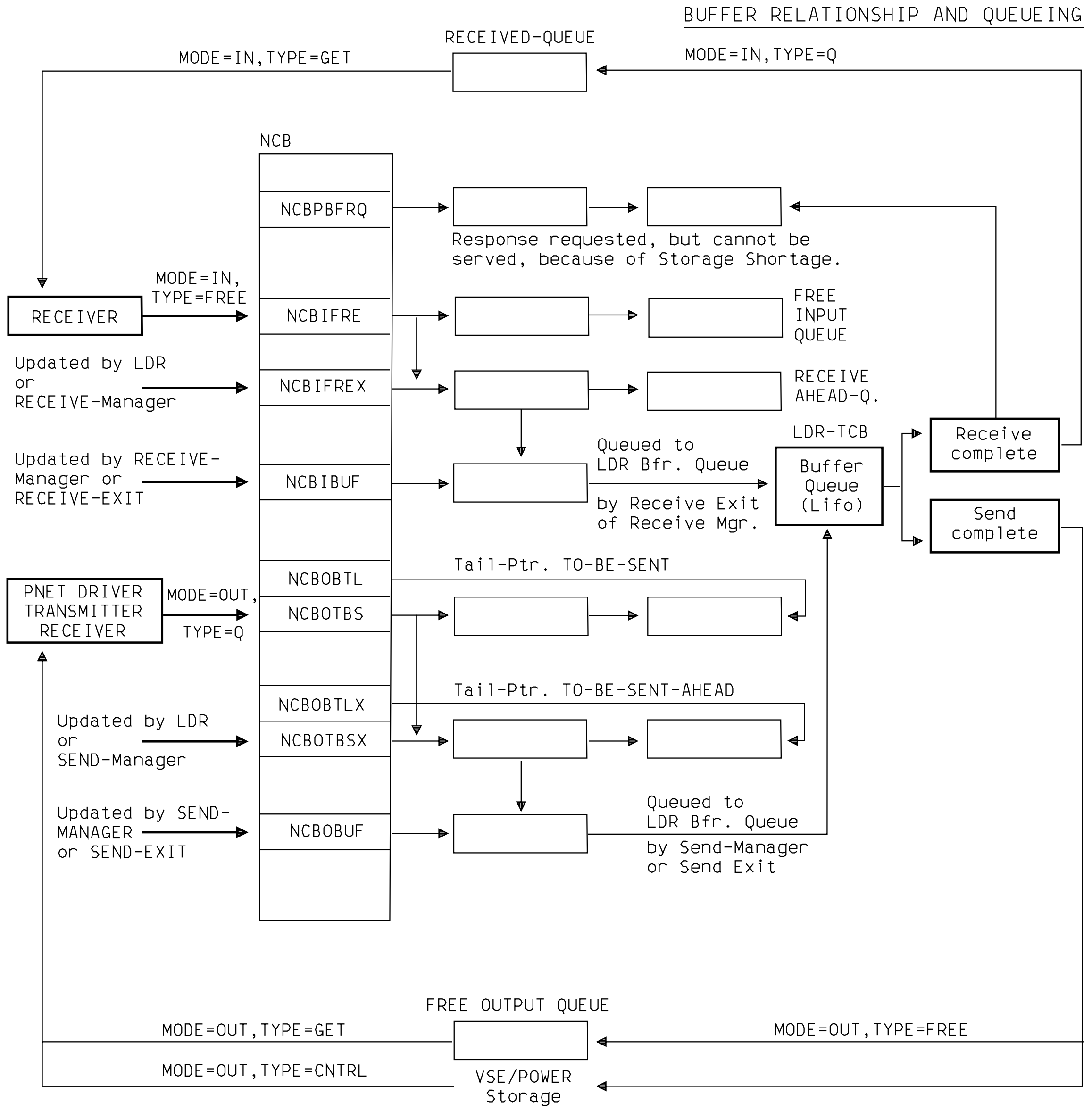


Figure 97. PNET SNA Buffer Relationship and Queuing

Remote Job Entry (RJE) Function

RJE,BSC

The VSE/POWER,BSC operations are performed by the following phases:

- IPW$$LM RJE BSC Line manager
- IPW$$BR RJE BSC Reader
- IPW$$BW RJE BSC Writer
- IPW$$BM RJE BSC I/O Monitor

At VSE/POWER initialization time, IPW$$I1 checks if RJE,BSC support is required, and if so, it saves the RJE part of the generation table.

Figure 98 on page 297 shows the relationship between the RJE,BSC tasks described in the following paragraphs.

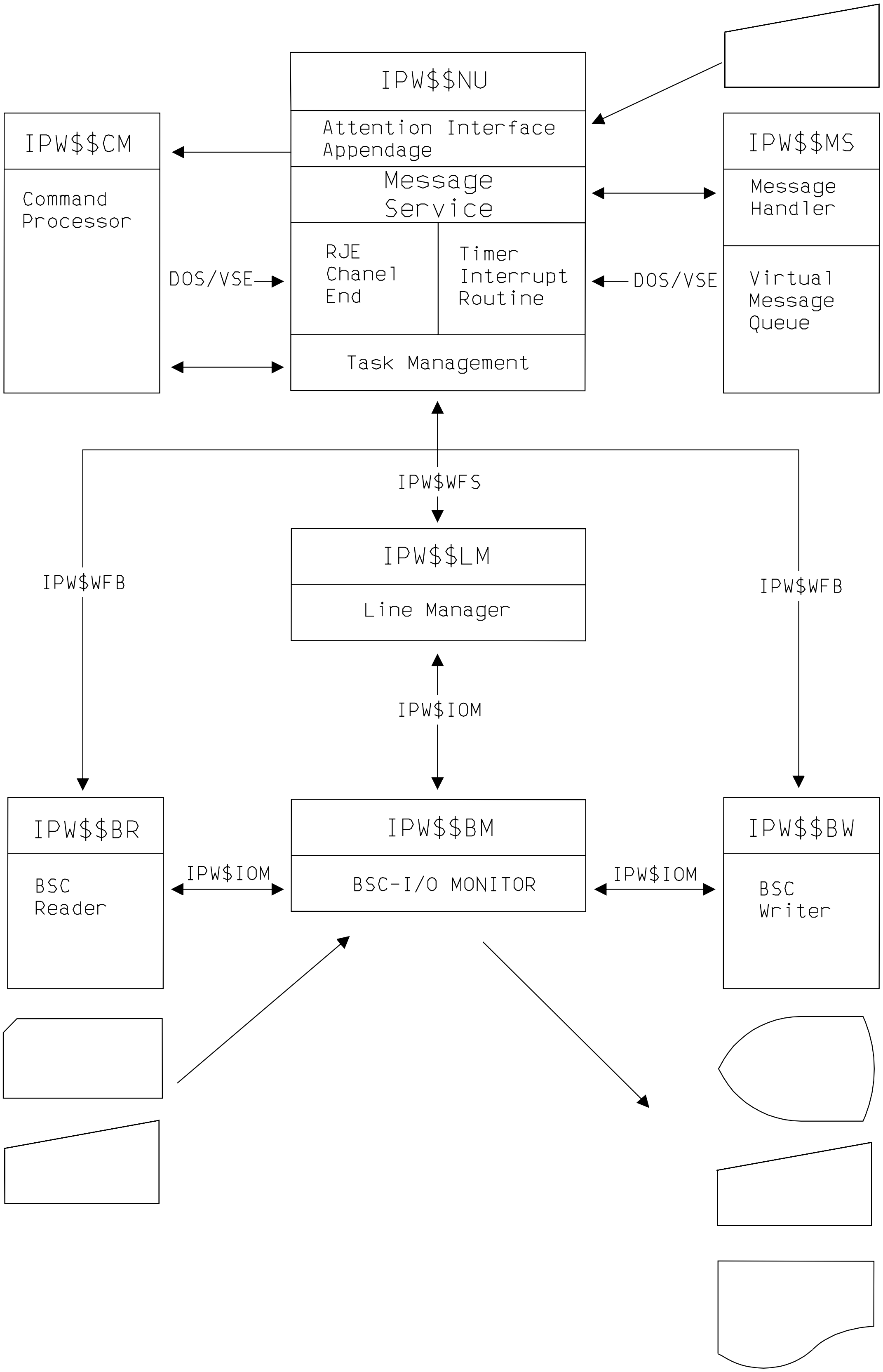


Figure 98. RJE,BSC Relationship

RJE,BSC Line Manager: The line manager task consists of three major functional areas:

1. Channel end processing

   including read,write, and control mode

2. Activity control

   including line initialization
   line start
   line close
   task creation
   task stop
   remote signoff processing

3. Line error handling with recovery

   including unit check recovery
   unit exception handling
   disastrous error handling

The task is activated, first by the command processor when the line is started or stopped, second from the RJE Channel End Appendage in the nucleus when the channel program completes by VSE/AF, or third by the RJE-BSC reader or writer task when the final signoff process should be initiated.

When the line manager is called it may test for any channel end which may have occurred while it was in the dormant state, or it may scan all LCBs of the system for any activity to be done. See Figure 99.

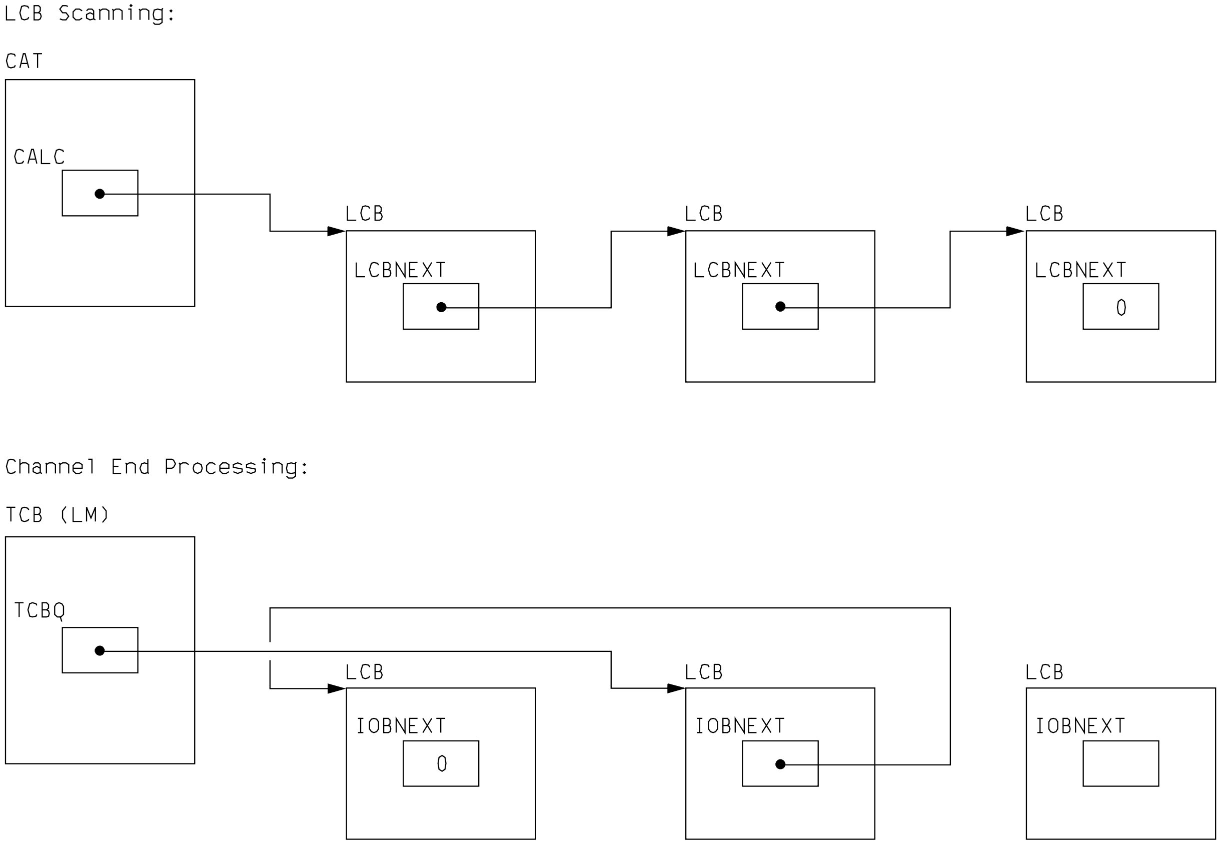


Figure 99. LCB Activity Checking and Channel End Processing

When a line start request is detected during LCB activity checking, two initial CCW chains are set up.

- Leased line
  - DISABLE
  - SET MODE
  - ENABLE
  - PREP
  - READ
- Switched line
  - DISABLE
  - SET MODE
  - ENABLE
  - NOP
  - READ

For leased line and switched line, the line is prepared to receive a response from the remote terminal and the line manager puts itself into the dormant state.

When a line stop/signoff request is detected during LCB activity checking, the line situation is tested for sending a remote message, a disconnect CCW chain is set up, and a decision to start the line again is made (see Figure 100).

LCB Indication	Switched Line	Leased Line
STOP	Line Stop	Line Stop
SIGNOFF	Line Start	Line Start

Figure 100. Line Action

The channel end routine consists of four sections:

- Control mode routine
- Receive mode routine
- Transmit mode routine
- Error routine

The Control Mode routine handles line initiation and line turnarounds from read to write mode or vice versa.

The Receive Mode routine checks the received data for valid BSC control characters or correct BSC framing control characters (start of text/ end of text/ end of block) and posts the appropriate reader task to process the received data. It calls the RJE,BSC Monitor to start another I/O request or to set a timer interval, waiting on a free buffer.

The Transmit Mode routine checks the acknowledgement or rejection response, frees the buffer if the buffer was acknowledged and posts the appropriate writer task. If the buffer was rejected, a retry is issued to send the buffer again.

When a line error is detected, this routine writes counter overflow records, unit check records, and end-of-day records to the VSE/AF recorder file on disk, and tries to recover from the error situation.

The line manager detaches itself only when a PEND command is given, and after all LCBs have been released.

RJE,BSC Reader: An RJE,BSC reader task is dynamically attached by the line manager when the terminal bids for the line by sending an ENQ. The task puts itself in the BSC wait state until it is posted by the line manager when the first block has been received. The data records are deblocked and expanded. An RJE,BSC command, read at job boundary is either processed by the RJE,BSC command processor routine or a temporary command processor is attached. Logical data records are passed to the logical reader. After the last record of the buffer received has been processed, the buffer will be made available to the line. If no software EOJ or hardware EOT has been read in the meantime, the task will wait for it. The next received block will be processed like the first one. After EOT on job boundary, the task detaches itself. If EOT is not detected on job boundary the task puts itself in the BSC wait state, waiting for real end-of-job.

RJE,BSC reader flow is illustrated in Figure 101.

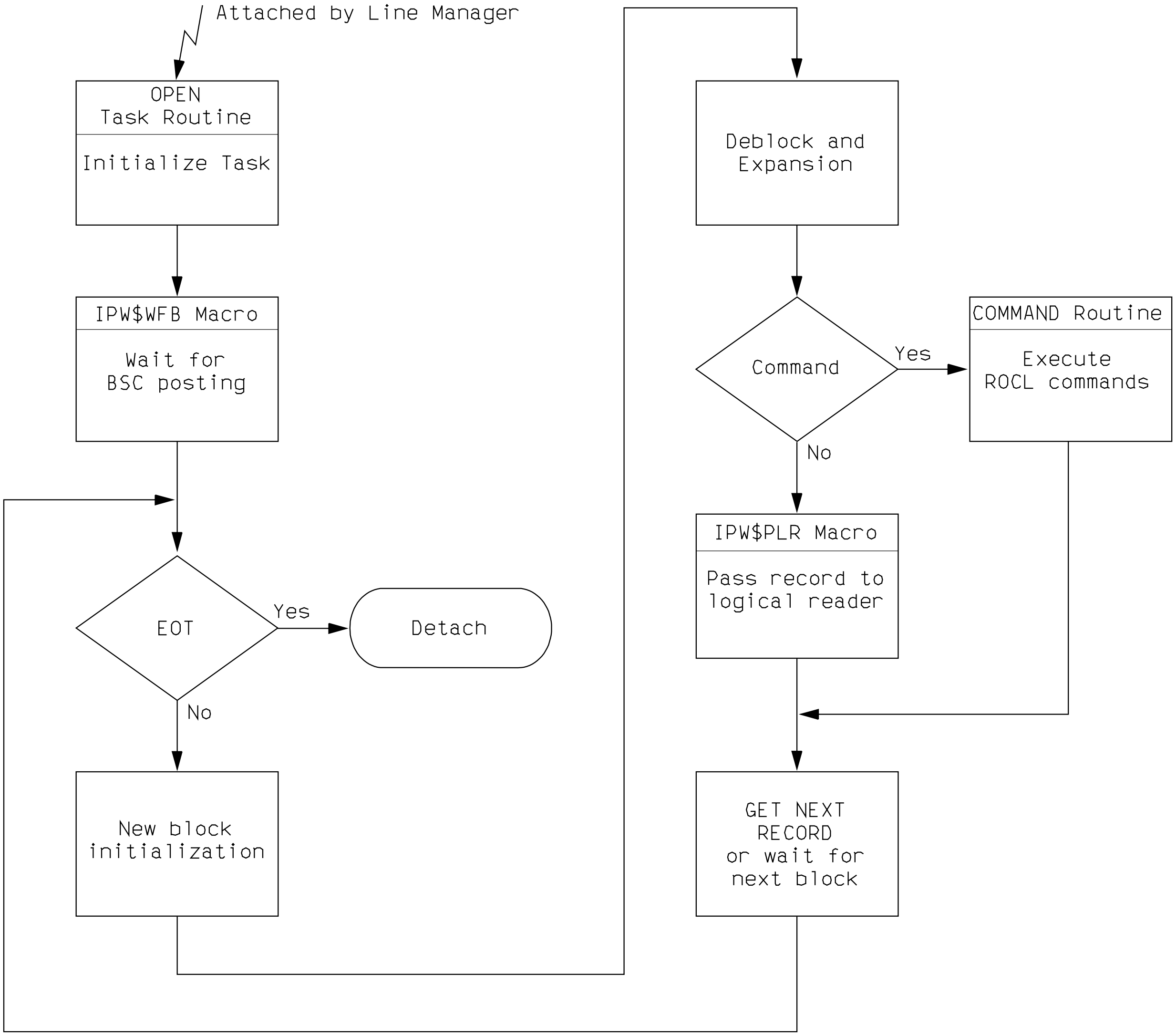


Figure 101. RJE,BSC Reader Flow

RJE,BSC Writer: An RJE,BSC writer task is dynamically attached by the line manager when

1. the remote operator has given the * .. START LST/PUN command.
2. LST/PUN output is available or becomes available and becomes ready for processing, and the * .. START LST/PUN command was previously issued by the remote operator.
3. One or more messages have been queued in the remote message queue.

If any messages are to be transmitted, the BSC writer task is called as a message task and no logical interface is opened.

If there is list or punch output to be transmitted, the logical interface to the logical writer is opened. Either logical data records from the logical writer or messages from the remote message queue are obtained and are grouped into physical data groups, after a positive response (ACK0) has been received to a line bid being sent by the line manager on behalf of the BSC writer task.

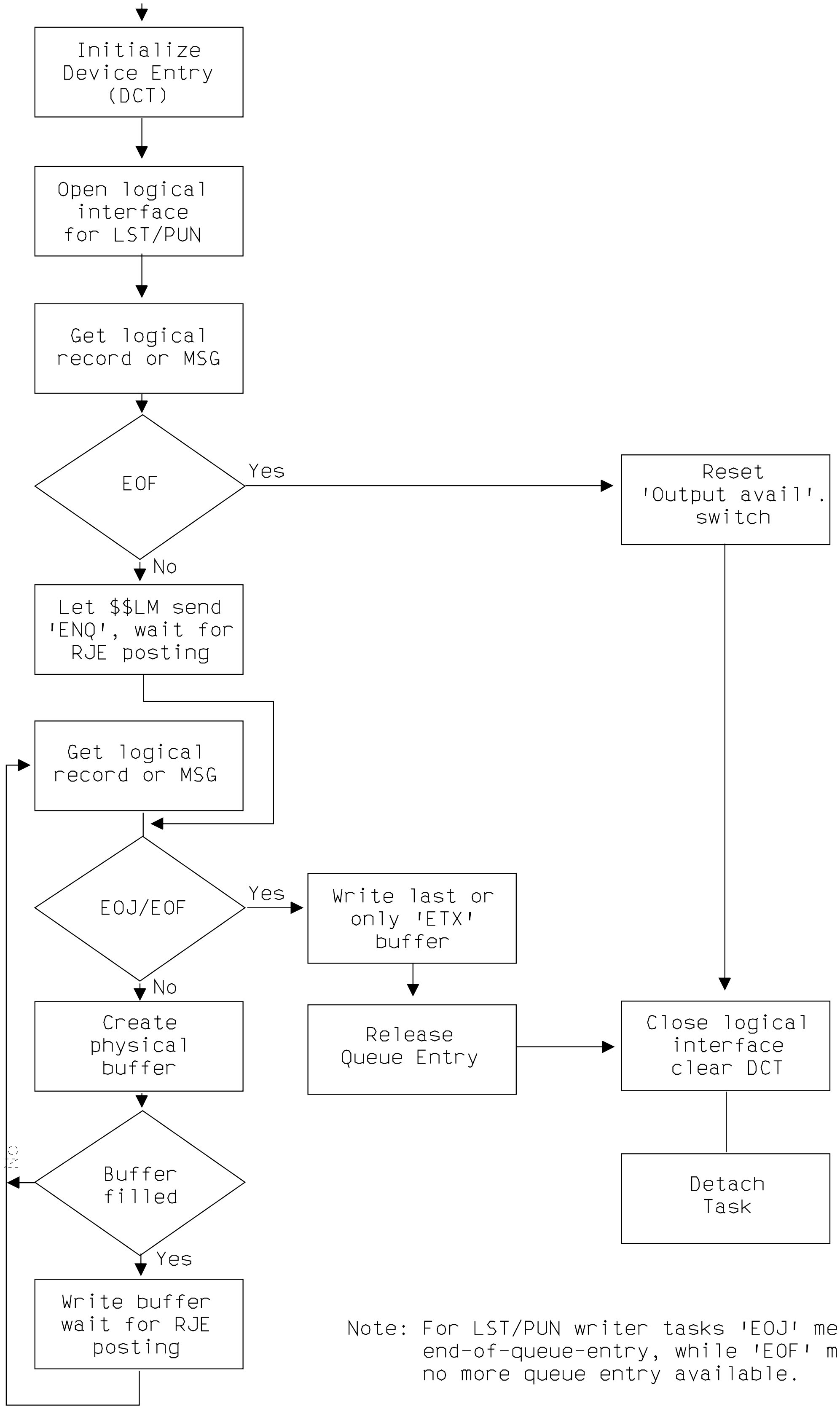
When grouped, the data is written out to the terminal by the BSC I/O Monitor at the time the next I/O operation is performed. At end-of-job the logical writer is again called to delete the queue entry from the queue according to its disposition. Then the BSC writer task detaches itself allowing the line manager to terminate the transmission by sending a EOT, if line turnaround was generated. The BSC writer task will then be re-attached, if output is still available.

If no more queue entry is available, the writer task indicates that an EOT has to be sent by the line manager after the last buffer has been sent.

When a forms change is required, the RJE,BSC writer task puts itself in the inactive state (IPW$WFO) awaiting remote operator response (SETUP or GO command) after the forms mount message is successfully transmitted to the remote station via a separate message task.

A BSC writer task detaches itself either at the end of a queue entry, or when all queue entries of the specified class(es) have been transmitted.

RJE,BSC writer flow is illustrated in Figure 102 on page 302.


Initialize Device Entry (DCT)
Open logical interface for LST/PUN
Get logical record or MSG
EOF
Yes
No
Reset 'Output avail'. switch
Let $$LM send 'ENQ', wait for RJE posting
Get logical record or MSG
EOJ/EOF
Yes
No
Write last or only 'ETX' buffer
Release Queue Entry
Close logical interface clear DCT
Create physical buffer
Buffer filled
No
Yes
Detach Task
Write buffer wait for RJE posting

Note: For LST/PUN writer tasks 'EOJ' means
end-of-queue-entry, while 'EOF' means
no more queue entry available.

Figure 102. RJE,BSC Writer Flow

RJE,BSC I/O Monitor: The BSC I/O Monitor handles the requests initiated for BSC line operations from the various BSC modules. Whenever an I/O operation is to be initiated, the Line Manager requests this by indicating the type of activity in the line control block and linking to the I/O Monitor.

On entrance, the request code being set is analyzed by dividing it into main and subrequest. The request code used to start I/O operations via SVC 0 is divided into the following categories:

- Line preparation and initiation
- Line turnaround from read to write mode and vice versa
- Read sequence
- Abnormal requests
- Write sequence
- Read only requests
- Retry

If the analysis results in a valid subrequest, the channel command words used to fulfill the request are created in the related line control block (LCB) and the I/O operation is started via SVC 0.

No WAIT is performed after SVC 0. Control is given back to the Line Manager immediately, which waits for I/O completion or processes asynchronous line activities.

The RJE,BSC I/O Monitor runs under control of the caller's TCB acting as a function module.

RJE,SNA

VSE/POWER RJE,SNA provides support for the SNA terminals that use Synchronous Data Link Control (SDLC). The communication with the SNA logical units is accomplished by using the VTAM access method. VSE/POWER controls the SNA work stations through a logical connection. All physical connections within the logical path are controlled by VTAM and NCP. Since VTAM does some of its processing under the TIK of the VSE/AF application task, the VSE/AF supervisor handles VTAM page faults as if they were VSE/POWER page faults. In order to minimize the effect of these page faults on non-RJE tasks, VSE/POWER attaches a VSE/AF subtask under whose PIB VTAM processing can be executed.

The VSE/POWER RJE,SNA operations are performed by the following phases:

- IPW$$SN (SNA manager)
- IPW$$LH (SNA logon processor 1)
- IPW$$LN (SNA logon processor 2)
- IPW$$IB (SNA inbound processor)
- IPW$$OB (SNA outbound processor)
- IPW$$OC (SNA outbound compaction manager)
- IPW$$MP (SNA message processor)
- IPW$$LF (SNA logoff processor)
- IPW$$VE (VTAM exit routines)

Figure 103 shows the VSE/POWER RJE,SNA interrelationship.

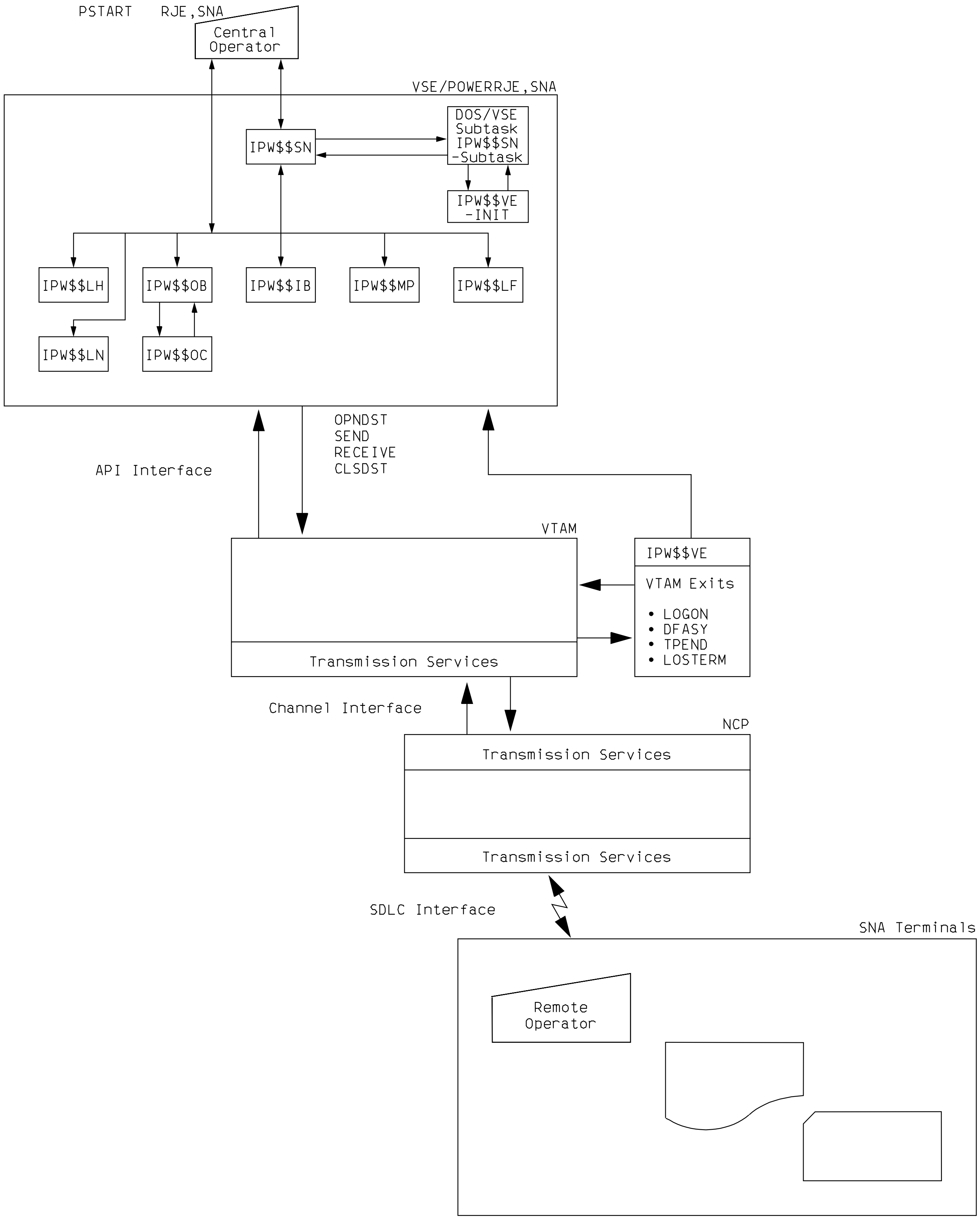


Figure 103. RJE,SNA Interrelationship

Initialization: When the central operator issues the PSTART command for SNA, the VSE/POWER SNA manager is attached to the TCB chain and controls on demand the activation of any inbound or outbound process related to a work station and its associated sessions. The SNA manager attaches a VSE/AF subtask to the VSE/POWER maintask in which the VSE/POWER application opens the interface with the VTAM access method by issuing the OPEN ACB macro. The ACB points to an EXLST control block, which defines the asynchronous exit structure within the VSE/POWER system to VTAM and consists of LOGON, LOSTERM, TPEND and DFASY exits. After the interface to VTAM is opened any logon request to VSE/POWER will be queued by VTAM. After the OPEN ACB request has been completed successfully VSE/POWER issues a SETLOGON START macro to enable VTAM to schedule the VSE/POWER LOGON exit routine.

Logon Processing: VTAM schedules the LOGON exit when a LOGON command is received from a logical unit. The LOGON exit routine queues the request.

In a Multiple Logical Unit (MLU) environment, VTAM and NCP do not associate sessions within a work station concept. VTAM and NCP only see individual sessions between VSE/POWER and the physical terminals. Hence, VSE/POWER is responsible for associating sessions with work stations according to the DATA operand of the LOGON command.

The first LOGON routine (IPW$$LH) in VSE/POWER processes all LOGON requests in the LOGON queue. For each LOGON request the routine performs the following functions:

- Utilizing a LOGON work area, it requests the user DATA and BIND parameters from VTAM issuing the INQUIRE SESSPARM macro.
- It performs syntax checking of the REMID and verifies its existence as specified in the PRMT macro.
- It checks the corresponding password if specified.
- It moves the 16 bytes of user information of the data into the session account record without validity checking.
- It verifies that the logical unit is authorized to log on with this REMID, provided that an LU=(name,...) parameter has been specified in the PRMT macro.
- It validates the BIND parameters.

In turn, the IPW$$LH routine checks whether or not any logical unit has been logged on with the same REMID. If no logical unit has as yet logged on with the same REMID, i.e., this work station is logging on its first session, the routine initializes all relevant work station and session related control blocks for this REMID.

If any logical unit has already logged on with the same REMID the routine verifies that this current LOGON request does not exceed the SESSLIM according to the specification in the PRMT macro. The routine initializes the relevant session-related control block, if SESSLIM is not exceeded. Else it rejects the LOGON request.

When the control blocks have been initialized the routine causes the second LOGON routine (IPW$$LN) to be attached, and it processes the next LOGON request from the queue.

The IPW$$LN routine completes processing the LOGON request for this session. It issues the OPNDST ACCEPT and SESSIONC SDT macros to VTAM and issues messages to the remote and central operators.

The LOGON request may be rejected for several reasons:

- Invalid user LOGON parameters, whereby VTAM sends a message to the central operator.

- System error (non-zero return code from VTAM).
- The BIND parameters are not accepted.
- The number of LOGON commands for a given work station exceeds the number specified by the SESSLIM parameter in the PRMT generation macro.
- The name of the logical unit logging on with a given REMID is not associated with the REMID specified in the PRMT macro.

  Note: This correlation of LU name and REMID is tested if and only if at least one LU name has been specified.
- The INQUIRE or OPNDST macro to VTAM was not successful.
- A GETVIS failed.
- The LOGON request provides an invalid REMID or password.

VSE/POWER then issues a CLSDST to VTAM resulting in a network procedure error being sent to the work station. Messages are sent to the central operator giving a LOGON reject reason code:

> '1V06I UNABLE TO LOGON luname RC=' or
>
> '1V26I INVALID REMID, PASSWORD, OR LUNAME RC='

The BIND Format: The BIND parameters exchanged between VSE/POWER and SNA terminals are detailed in Figure 104 on page 307. If presentation services (PRSVC) are not provided by VTAM (byte 14=0), default values for bytes 14-26 will be forced. VSE/POWER is flexible in its BIND requirements. Each BIND parameter affords one of the following characteristics:

- Ignored parameter (I)
- Enforced parameter (E)
- Mandatory parameter (M)
- Variable parameter (V).

Ignored parameters are neither tested nor modified by VSE/POWER. Enforced parameters are dictated by VSE/POWER on the secondary logical unit. Mandatory parameters are tested, and if incorrect, the LOGON request is rejected. The variable parameters are copied and affect processing. For a coding example, refer to Appendix A in the *VSE/POWER Remote Job Entry* , which describes LOGON Mode Table and BIND parameter requirements.

Bytes	Bits	Content	Description
0	0–7	X'31'	BIND RU code
1	0–3	B'0000'	Format: 0 (M)
	4–7	B'0001'	Bind type non–negotiable. (E)
2		X'03'	FM profile 3. (M) (i.e. Immedicate Resp. Mode ...)
3		X'03'	TS profile 3. (M)
Primary NAU Protocol			
4	0	B'1'	Multiple RU per chain. (M)
	1	B'0'	Immediate request mode. (M)
	2–3	B'10'	Definite response chains. (M)
	4–5	B'00'	Reserved. (E)
	6	B'0'|B'1'	Compression may not be | may be used. (V)
	7	B'1'	Primary may send EB. (M)
Secondary NAU Protocol			
5	0	B'1'	Multiple RU per chain. (I)
	1	B'0'	Immediate request mode. (M)
	2–3	B'10'	Definite response chains. (M)
	4–5	B'00'	Reserved. (E)
	6	B'0'	Compression may not be used. (E)
	7	B'1'	Secondary may send EB. (M)
Common LU Protocol			
6	0	B'0'	Reserved. (E)
	1	B'1'	FM headers may be used. (M)
	2	B'1'	Brackets are used for PLU and SLU. (M)
	3	B'1'	Bracket termination rule 1. (M)
	4	B'0'|B'1'	Alternate code may not be used | may be used (ASCII). (V)
	5–7	B'000'	Reserved. (E)

Figure 104 (Part 1 of 3). BIND Format

Bytes	Bits	Content	Description
7	0–1	B'10'	HDX flip–flop. (M)
	2	B'0'	Primary responsible for ERP. (M)
	3	B'0'	Secondary is first speaker. (M) (Contention winner)
	4–6	B'000'	Reserved. (E)
	7	B'0'	Reserved. (E)
TS Usage (TSU)			
8	0	B'0'	Staging indicator – 1 stage enforced (E)
	1	B'0'	Reserved. (E)
	2–7	B'000000'	SLU send pacing count. (I)
9	0–1	B'00'	Reserved. (E)
	2–7	B'000000'	SLU receive pacing count. (I)
10		X'85'	SLU max RU size (256 bytes). (E)
11		X'85'	PLU max RU size (256 bytes). (E)
12	0	B'0' | B'1'	Staging indicator (1 stage | 2 stage) (I)
	1	B'0'	Reserved. (E)
	2–7	B'000000'	PLU send pacing count. (I)
13	0–1	B'00'	Reserved. (E)
	2–7	B'000000'	PLU receive pacing count. (I)
Presentation Services (PRSVC)			
14		X'01'	LU session type (if = 00, set default = 01; if ” 00, M = 01)
15	0–3	X'1'	FMH header set = 1 (E)
	4–7	X'0'	SCS Basic. (E)
PLU Usage Indication (Outbound)			
16	0	B'0'	Interrupt (no. of levels). (I, def. =B'0')
	1	B'0'|B'1'	No compaction | compact. (V, def. =B'0')
	2	B'0'|B'1'	No PDIR | PDIR. (V, default =B'0')
	3–7	B'00000'	Reserved. (E)
17		X'00'	Reserved. (E)

Figure 104 (Part 2 of 3). BIND Format

Bytes	Bits	Content	Description
18	0	B'1'	BS, CR, INP, ENP, LF, HT, VT allowed. (M)
	1	B'1'	SHF allowed. (M)
	2	B'1'	SVF allowed. (M)
	3	B'1'	SVF, SEL allowed. (M)
	4–6	B'000'	Reserved. (E)
	7	B'1'	TRN, IRS allowed. (M)
19		X'00'	Reserved. (E)
20	0	B'0'|B'1'	Document output not allowed | allowed. (V, default = B'1')
	1	B'0'|B'1'	Card format not allowed | allowed. (V, default = B'1')
	2	B'0'	Exchange media not allowed. (E)
	3	B'0'	Disk, data management not allowed. (E)
	4–7	X'0'	Reserved. (E)
SLU Usage Indication (Inbound)			
21	0	B'0'	Interrupt (number of levels). (E)
	1	B'0'	No compaction. (E)
	2	B'0'	No PDIR. (E)
	3–7	B'00000'	Reserved. (E)
22		X'00'	Reserved. (E)
23	0	B'0'	BS, CR, INP, ENP, LF, HT, VT not allowed. (E)
	1	B'0'	SHF not allowed. (E)
	2	B'0'	SVF not allowed. (E)
	3	B'0'	SVF, SEL not allowed. (E)
	4–6	B'000'	Reserved. (E)
	7	B'1'	TRN, IRS allowed. (I, default = B'1')
24		X'00'	Reserved. (E)
25	0	B'0'	Document output not allowed. (E)
	1	B'1'|B'0'	Card fmt allowed | not alld. (V,def.=B'1')
	2	B'0'|B'1'	Exchange media not allowed | allowed (V)
	3	B'0'	Disk, data management not allowed. (E)
	4–7	X'0'	Reserved. (E)
26		X'00'	Reserved. (E)

Figure 104 (Part 3 of 3). BIND Format

Host Workstation Communication: Logical records are grouped into RUs which are logically grouped into chains. Output related to one VSE/POWER queue entry (job or segment) is sent as one chain unless interrupted by an inbound flow or an outbound message. An outbound job is always sent as a DS (data stream). Messages are sent as an only-chain.

Input related to one VSE/POWER queue is not related to a chain by VSE/POWER. VSE/POWER only identifies job boundaries according to VSE/POWER JECL or VSE/AF JCL statements with the exception that an end bracket forces End of JOB (EOJ), if no valid VSE/POWER job delimiter was found. It is the option of the work station to associate jobs and chains if this association simplifies ERP (error recovery procedures) at the work station.

VSE/POWER supports all three SNA function management headers for outbound, that is, FMH1, FMH2 and FMH3, but only FMH1 for inbound. Concatenation of FMHs is not supported. If VSE/POWER receives an FMH with the concatenation bit on, it returns an exception response.

Function Management Header Type 1 (FMH1): VSE/POWER supports the standard 6-byte FMH1 for device selection and delimiting data set operations. Refer to Figure 105 on page 311 for details of the FMH1 format layout.

Bytes	Bits	Name	Content	Description
0	0–7		X'06'	Length of FMH1
1	0	FMHC	B'0'	Concatenation not supported
	1–7	TYPE	B'0000001'	Type 1 FMH
2	0	DEMAND		
		SELECT	B'0'	Ignored
	1–3	MEDIA	B'000'	CONSOLE
			B'001'	EXCHANGE MEDIA (only inbound)
			B'010'	CARD
			B'011'	PRINT
				All other codes not supported.
	4–7	LOGICAL	X'0'	1st logical device
		ADDRESS	X'1'	2nd logical device for print
			X'2'	3rd logical device data only
3	0	STACKREF		Stack reference indicator
			B'0'	Refers to DS begun by sender
			B'1'	Refers to DS begun by receiver
	1–3		B'000'	Reserved
	4–7		B'0000'	Data Stream Profile (DSP)
				X'000'=default implied by Media (byte 2)
4		PROPERTY		
	0–2		(See Note)	DS selection
	3	DST	B'0'	Basic exchange not supported
			B'1'	Basic exchange supported (inbound only)
	4		B'0'	Reserved
	5	CMI	B'0'	No compression
			B'1'	Compression (outbound print only)
	6	CPI	B'0'	No compaction
			B'1'	Compaction (outbound print only)
	7		B'0'	Reserved
5	0–7	ERCL	X'00'	Basic exchange record length must be ◄= 128
6–n				DSNAME which is defined by architecture in bytes 6–n is not supported by VSE/POWER.

Figure 105. FMH1 Format

Note: The data stream selection bits are used in combination. The valid combinations are:

B'000' - Resume suspended data stream (RDS)
B'001' - End current data stream (EDS)
B'010' - Begin data stream (BDS)
B'011' - Begin and end data stream (BEDS)
B'100' - Suspend current data stream (SDS)
B'101' - Abort (abnormally end) current data stream (ADS)
B'110' - Reserved
B'111' - Reserved.

The following should be noted:

1. With the resumption of a suspended outbound data stream, VSE/POWER will not change any of the FMH options selected in the original FMH.
2. An FMH may exist in an RU only at first-of-chain (FC) or only-chain (OC). The presence of an FMH is signaled by the format indicator bit in the request header. If data is received with no FMH where an FMH is expected, the default FMH applies as in Figure 106.
3. When the data stream selection bits are set to B'011' the entire data stream is being sent within one chain, including the FMH. Print and card media output data are initiated by only-chain FMHs indicating BDS, followed by chain(s) of data, and terminated by an only-chain FMH indicating EDS.
4. An FMH1 (BDS) is sent prior to, and an FMH1 (EDS) after, each job output or segment. FMH1 (BDS) is sent after FMH3.

Bytes	Bits	Name	Content	Description
0	0–7	Length	X'06'	Length of FMH1
1	0	FMHC	B'0'	Concatenation not support
	1	reserved	B'0'	
	2–7	TYPE	B'000001'	Type 1 FMH
2	0	DEMAND SELECT	B'0'	Ignored
	1–3	MEDIA	B'000'	CONSOLE
	4–7	LOGICAL ADDRESS	X'0'	1st console
3	0	STACKREF	B'0'	Stack reference indicator
	1–7		B'0000000'	
4	0–2	DS sel.	B'011'	Begin and end of data stream
	3	DST	B'0'	Basic exchange not supported
	4	reserved	B'0'	
	5	CMI	B'0'	No compression
	6	CPI	B'0'	Compaction not supported
	7	reserved	B'0'	
5	0–7	ERCL	X'00'	Basic exchange record length

Figure 106. Default FMH1

Function Management Header Type 2 (FMH2): The FMH2 represents the peripheral data set information record (PDIR). It carries information relative to the destination selected by FMH1. VSE/POWER only supports FMH2 outbound, but not inbound.

The format of the FMH2 is shown in Figure 107.

Bytes	Bits	Name	Content	Description
0	0–7	Length	X'44'	Length of FMH2
1	0	FMHC	B'0'	No concatenation
	1			Reserved
	2–7	TYPE	B'000010'	FMH type 2
2		CODE	X'01'	PDIR
3	0–7	Identif.	X'00'	Ordinary data set
4–11		DATE	MM/DD/YY	Date of queue entry creation in the form MM/DD/YY.
12–19		TIME	HH.MM.SS	Time of queue entry creation in the form HH.MM.SS.
20–27		FORMS	C'ccccbbbb'	Forms name in the form C'ccccbbbb' Default is all blanks. The forms can be provided by the * $$ LST FNO= parameter, by the LFCB macro, or by the SEGMENT macro, JECL= operand, where an * $$ LST FNO= is specified.
28–35		FCB	C'cccccccc'	FCB name (1–8 characters)in the form C'cccccccc' left justified. Default is all blanks. The FCB can be provided by the * $$ LST FCB= parameter, by the LFCB macro, or by the SEGMENT macro, JECL= operand, where an * $$ LST FCB= is specified.
36–43		TRAIN	C'bbbbbbbb'	Not supported
44–51		COPIES	C'cccccccc'	Copies (1–8 characters). Indicates the number of additional copies, i.e., zero means one copy. EBCDIC characters (digits), right justified, without leading zeroes, except low order digit. The maximum number allowed is 98. The The number of additional copies plus one can be provided by means of the PALTER command, or it can be provided by the * $$ LST COPY= parameter, or by the SEGMENT macro, JECL= operand, where a * $$ LST COPY= is specified.
52–59		VOLIO	C'cccccccc'	Volume of I/O in the form C'cccccccc' right justified with leading zeros suppressed. If printer selected the field indicates the number of expected print lines.
60–67		JOB NAME	C'cccccccc'	Job name in the form C'cccccccc' left justified. The job name can be provided by the * $$ JOB JECL statement. Default is AUTONAME.

Figure 107. FMH2 Format

The FMH2 is sent as an only-chain in DS state after FMH1 has been sent, provided that the PDIR bit in the BIND parameters was turned on at logon time of the session. If the PDIR bit is off the SETUP/GO procedure will be performed.

The PDIR is always sent if the BIND indicates so, regardless of whether or not forms change is required. Without PDIR indicated in BIND the SETUP/GO procedure is performed only if forms change is required.

Function Management Header Type 3 (FMH3): The FMH3 carries information relative to the entire session. Type 3 information applies to all destinations reached through this session. The FMH3 is sent as only-in-chain and it is not chained with another FMH, nor does the RU contain any other data. VSE/POWER supports only outbound FMH3. If VSE/POWER receives an inbound FMH3 it returns an exception response.

The format of the FMH3 is shown in Figure 108.

Byte	Bits	Name	Content	Description
0	0–7	Length	(See Note)	Length of FMH3
1	0	FMHC	B'0'	No concatenation
1	1	Reserved		
1	2–7	TYPE	B'000011'	Type 3 FMH
2	0–7	FUNCTION	X'02'	Compaction table
3	0–7	MASTER NO	3–16	No. of master characters
4–n		TABLE	(See Note)	Compaction table characters

Figure 108. FMH3 Format

Note: Length is dependent on length of the compaction table. It can be calculated by:

```
length = 4 + 256 - (m x m) for m < 16
length = 4 + 16 for m = 16
```

where m is the number of master characters.

The FMH3 including length indication is generated by the PCPTAB macro.

An FMH3 is sent to the secondary logical unit whenever a job is to be transmitted outbound in compacted form using a compaction table other than the one currently valid for the session. The FMH3 is always sent as only-chain, without data, and between DS state. The FMH3 RU may or may not indicate begin bracket depending on whether or not the session is already in bracket state.

Initiation of Data Processing

Data Inbound Processing: An inbound processor task is attached for a given session by the SNA manager in the following cases:

- A VTAM RECEIVE ANY is satisfied: The SNA manager determines the session on which an inbound flow is to be expected by means of a pointer in the user field of the RPL. It then attaches an inbound processor, and reissues RECEIVE ANY to allow input from other sessions to be accepted.
- An inbound flow is interrupted for an inbound message: The inbound processor being interrupted posts the SNA manager which attaches a second inbound processor for this session.
- An outbound flow is interrupted for an inbound flow or message: The outbound processor being interrupted posts the SNA manager which attaches an inbound processor for this session and reissues RECEIVE ANY.

In all three cases the inbound processor issues RECEIVE SPECIFIC. It verifies whether or not the device (RDR1 or console) selected by FMH1 (implicitly or explicitly) is already in use. If in use it rejects the inbound flow.

Data Outbound Processing: An outbound processor task is attached for a given session by the SNA manager in the following cases:

- Outbound Data. When a job is available in an output queue (list or punch) of a given class with a given REMID, the queue management (IPW$$AQ) routine of VSE/POWER scans the control blocks for a match of the REMID. When the REMID is found the routine scans the classes of all outbound devices for this REMID. These classes are assigned to the devices by means of the START command. When a match is found, and if the device has been started, the routine flags the device and posts the SNA manager.

  The SNA manager finds the flagged device and searches for a free session. If a free session is found the SNA manager attaches an outbound processor which starts processing the job output until the output queue is empty. When the queue is empty the outbound processor resets the device flag, posts the SNA manager and detaches itself.

  The SNA manager does not take further action if no free session is found. It will repeat the attempt when it is posted again, for example, when a processor is detached.

- Outbound Message. Outbound messages are queued by the message service routine (IPW$$MS). Whenever the routine queues a message for a given REMID it posts the SNA manager. The SNA manager searches a free session to the work station identified by the REMID. If a free session is found the SNA manager attaches the message processor which sends the message to the work station.

  If no free session is found the SNA manager searches a session involved in an outbound flow. The search begins for a session which is waiting for a GO command or RESTART following intervention required. If not found, then the search continues for some session which is transmitting. If found, it flags the associated control block which causes the outbound processor to suspend. Upon suspension the SNA manager attaches the message processor.

  No action is taken by the SNA manager if no session involved in an outbound flow is found. The SNA manager will repeat the attempt when it is posted again, for example, when a processor is detached.

Once attached, the message processor transmits all messages queued for a given REMID and detaches.

Interruption of Data Processing

Interruption of Data Outbound: The interruption of the outbound processor can be caused by the following conditions:

- A SIGNAL from the work station has been received.

  The outbound processor forces the termination of the current chain, sends an FMH1 with suspend DS and a change direction indication to the logical unit. It then posts the SNA manager.

- A message is pending.

  The outbound processor forces the termination of the current chain, sends an FMH1 with suspend DS to the logical unit, and posts the SNA manager which attaches the message processor.

Interruption of Data Inbound: Interruption of an inbound processor receiving card data is accepted anytime when the logical unit has a message to send.

The interruption must be indicated to the inbound processor by an inbound FMH1 with suspend DS. The suspended inbound processor will then post the SNA manager which will attach a second inbound processor to receive messages. These messages are treated as commands. Upon reception of an FMH1 with resume DS, the interrupting inbound processor will detach, the suspended inbound processor will resume the DS, and normal inbound flow can continue.

Inbound interruptions for outbound data are not supported.

Protocols: Half-duplex, flip-flop mode protocols are employed. Only one data stream at a time is allowed per session and contention is resolved by the use of SNA brackets.

Termination

Session Termination: The termination of a session is requested by the remote operator either by issuing the LOGOFF request through VTAM, or by submitting a SIGNOFF command (from card or via the console) in the inbound data flow. The LOGOFF request may be an unconditional LOGOFF in which VTAM breaks the session and notifies VSE/POWER through the LOSTERM exit. If the remote operator issues a conditional LOGOFF VTAM notifies VSE/POWER also through the LOSTERM exit, but VTAM does not break the session. The SIGNOFF command is passed via the normal inbound data stream directly to VSE/POWER where it is handled as a conditional LOGOFF request for all sessions of a given work station.

The work station may log off any individual session within the MLU concept. The LOGOFF may be conditional or unconditional. The SIGNOFF command causes LOGOFF of all sessions of the work station conditionally.

VSE/POWER handles the unconditional LOGOFF as an emergency stop which means that the termination routines are entered without checking any internal job boundary state. In this case the current reader job entry will not be added to the queue. The conditional LOGOFF will be interpreted as a request for an orderly deactivation of the current session. In this case the termination routines will be entered only at job boundaries, that is, when processing of the current job entry is completed.

After the active processors have been terminated, either normally or abnormally, the SNA manager activates the LOGOFF processor which sends a message to the work station and finally issues a CLSDST to terminate the session. In an MLU environment a SIGNOFF causes termination session-by-session at job boundary times.

Session termination can be caused by the central operator either by means of the PSTOP command or, in case of emergency, by issuing the VARY NET,INACT,I,ID=luname command. VTAM notifies VSE/POWER in the LOSTERM exit. Because VTAM does not allow any I/O request to be issued, VSE/POWER handles this termination type similar to an unconditional LOGOFF.

Application Termination: The central operator may cause RJE,SNA shutdown either through VSE/POWER central operator commands (for example, PSTOP or PEND) or through VTAM operator commands (for example, HALT). Refer to *VSE/POWER Installation and Operations Guide*.

RJE,SNA Routines: Figure 109 on page 317 briefly describes each of the routines used to support RJE,SNA. Figure 110 on page 320 further describes the control blocks and work areas used to aid execution. In addition, Figure 112 on page 330 illustrates the scheme of chaining the control blocks.

An overview of the sequence of routine execution and events is provided by Figure 111 on page 321. This figure should be used along with Figure 109 on page 317 to understand the RJE,SNA architecture.

Routine	Called/ Attached by	Returns to	Function or Notes
IPW$$IB Inbound Processor	IPW$$SN	IPW$$NU	Issues RECEIVE Specific request to VTAM to receive data and then deblocks the data for spooling by IPW$$LR. Processes remote operator commands: • START • STOP • FLUSH • RESTART • SETUP • GO • SIGNOFF and transfers all other commands to IPW$$CM for processing. Posts the outbound processor command following GO, or RESTART when intervention is required. Posts the SNA manager and detaches.
IPW$$LF Logoff Processor	IPW$$SN	IPW$$NU	Logs off a logical unit using the VTAM macros SESSIONC and CLSDST. Sends message "1V12I" to the remote terminal and then sends the central operator the message "1V11I". Posts the SNA manager and detaches.
IPW$$LH Logon Processor 1	IPW$$SN		Establishes SNA control block construction (SUCB, LUCB and WACB). Checks LOGON request and LU BIND parameters for validity. Sets an indicator for IPW$$SN to attach logon processor No. 2 Posts the SNA manager and detaches.
IPW$$LN Logon Processor 2	IPW$$SN	IPW$$NU	Establishes SNA session and starts data traffic with VTAM macros OPNDST and SESSIONC. Prints message "1V09I REMOTE rrr LOGGED ON TO POWER ON luname" at central operator console and then queues the same message for the remote terminal to be sent by the message processor (IPW$$MP). Posts the SNA manager and detaches.

Figure 109 (Part 1 of 3). Description of RJE,SNA Routines

Routine	Called/ Attached by	Returns to	Function or Notes
IPW$$MP Message Processor	IPW$$SN	IPW$$NU	Transmits messages in message queue using VTAM macro SEND. Posts the SNA manager and detaches.
IPW$$OB Outbound Processor	IPW$$SN as LSTN or PUN task	IPW$$NU	Obtains job output data from spool file and transmits data to the LU using the VTAM macro SEND. The following intermediate steps occur: • Obtain spool file through IPW$$LW. • Create Function Management Headers (FMH). • Compress and compact if required • Pack data into request units(RU) Waits on GO posting from IPW$$IB if SETUP remote command issued. Posts the SNA manager and detaches.
IPW$$OC Outbound Compaction	IPW$$OB	IPW$$OB	Creates and updates COCB(s) and loads compaction table phases into GETVIS area.
IPW$$SN SNA Manager	IPW$$CP	IPW$$NU	Sets up following ECBs in the TCB of IPW$$SN: • VTAM RECEIVE ANY ECB • Work ECB for RJE,SNA posting of IPW$$SN. Attaches a VSE subtask. Issues VTAM "RECEIVE ANY" macro. Prints central operator message "1V04I" and waits on ECB posting.
IPW$$SN Segment SUBTASK	IPW$$SN INIT	DOS/VS	When called the first time at RJE,SNA startup time, it calls the IPW$$VE—Segment INIT and enables communication through VTAM with SETLOGON macro. Then posts IPW$$SN ECB and VSE/POWER master ECB, and waits on posting by IPW$$SN. At termination time, the VTAM macro SETLOGON QUIESCE is called to halt further LOGON requests.

Figure 109 (Part 2 of 3). Description of RJE,SNA Routines

Routine	Called/ Attached by	Returns to	Function or Notes
IPW$$SN Segment MAIN	Posted by: • VTAM due to: RECEIVE ANY input via VTAM exit • VSE/POWER routines: IPW$$AQ IPW$$CM IPW$$IB IPW$$LH IPW$$LN IPW$$MS IPW$$OB	IPW$$NU	After VTAM posting due to SNA line activity, a RDR task is attached which causes IPW$$IB to execute. After posting from other VSE/POWER routines, a scan of the work station control blocks (SUCBs) and logical unit control blocks (LUCBs) is made. If any found to be active, the appropriate processor tasks are attached: • LST or PUN tasks (IPW$$OB) • Messages (IPW$$MP) • Logon/logoff tasks Then a loop back is made to wait on further posting.
IPW$$SN Segment TERM	IPW$$SN MAIN	IPW$$NU	Frees certain work areas and control blocks. Prints message "1V05I". Detaches IPW$$SN task.
IPW$$VE Segment LOGON	VTAM	VTAM	Creates and chains control blocks, used at logon time. Posts IPW$$SN work ECB and VSE/POWER master ECB.
IPW$$VE Segment INIT	IPW$$SN SUBTASK	IPW$$SN SUBTASK	Inserts addresses of VTAM exits in the ACB exit list.
IPW$$VE Segment DFASY	VTAM	VTAM	If request to interrupt data flow, then the signal received indicator is set in the LUCB of the LU. If request to shutdown, then stop session indicator is set. Posts IPW$$SN work ECB and VSE/POWER master ECB.
IPW$$VE Segment TPEND	VTAM	VTAM	Sets SNA stop code in SNCB. Posts IPW$$SN work ECB and VSE/POWER master ECB.
IPW$$VE Segment LOSTERM	VTAM	VTAM	Sets on the stop session indicator in the LUCB of the LU. Posts IPW$$SN work ECB and VSE/POWER master ECB.

Figure 109 (Part 3 of 3). Description of RJE,SNA Routines

RJE,SNA Control Blocks/Work Areas	When		Created by:	Stored at:	Purpose/General Contents
	Created:	Freed:			
CAT (Control Adress Table)	At VSE/POWER Initialization	At VSE/POWER Termination	IPW$$I1	Real storage area	Pointers to modules and major control blocks
SNCB (SNA Control Blockse)	At VSE/POWER Initialization	At VSE/POWER Termination	IPW$$I7	Fixed real storage area	RJE,SNA control block.
COCB (Compaction Control Block)	At time of first compaction table usage	At RJE,SNA Termination (IPW$$SN)	IPW$$OC	GETVIS area	Contains compaction table names, pointer and status.
LRCB (Logon Request Control Block)	At first LU LOGON	When (last) LOGON processed (IPW$$SN)	IPW$$VE	GETVIS area	Used for LOGON processing. Consists of a neader plus LRUBs
LUCB (LU Control Block)	Space reserved at logon of first LU of work station Initialized by IPW$$LN	At work station logoff time IPW$$LF	IPW$$LH	GETVIS area	Contains information required for LU session, e.g.variable BIND parameter information
Logon LUCB	At LOGON of first LU of work station	At RJF,SNA Termination (IPW$$SN)	IPW$$VE-Segment LOGON	GETVIS area	Contains information required for LU session. Used as work area during logon processing.
SUCB (SNA UNIT Control Block)	Space reserved at LOGON of first LU of work station	At LOGOFF OF LAST LU of work station (IPW$$LF)	IPW$$LH	GETVIS area	Contains information pertaining to the work station of two types: a) General information, for example: REMID, SESSELIM . b) Cevice information for LSTn, LSTn, PUN, RDR and Console for example: class. Created by copying the LOGON SUCB onto the reserved SUCB AREA> One is created for each work station logged on.
Logon SUCB	At logon of first LU of work station	At RJE,SNA Termination IPW$$SN	IPW$$VE-Segment	GETVIS area	Used as a work area during LOGON processing.
WACB	* WACB for inbound interruption * WACB for LU inbound data * WACB for LU outbound data	* At LOGOFF of last session of work station IPW$$LF * At LOGOFF of LU IPW$$LF * LSTn/ PUN task termination (MP,OB)	IPW$$LH IPW$$LH IPW$$OE IPW$$MP	GETVIS area GETVIS area GETVIS area	Contains VTAM RPLs, RU buffers and some BIND information. One exists for each SIUB; another exists for each LUCB logged on for inbound; and another exists for each LUCB logged on during outpound data on message handling.
LOGON WACB	At LOGON of first LU of work station	At RJF,SNA Termination (SN)	IPW$$VE-Segment	GETVIS area	Used as work area for LOGON processing
RMCB (Remote Control Block)	VSE/POWER Initialization	VSE/POWER Termination	IPW$$I7	GETVIS area	Contains information from the PNMT macro.

Figure 110. Description of RJE,SNA Control Blocks and Work Areas

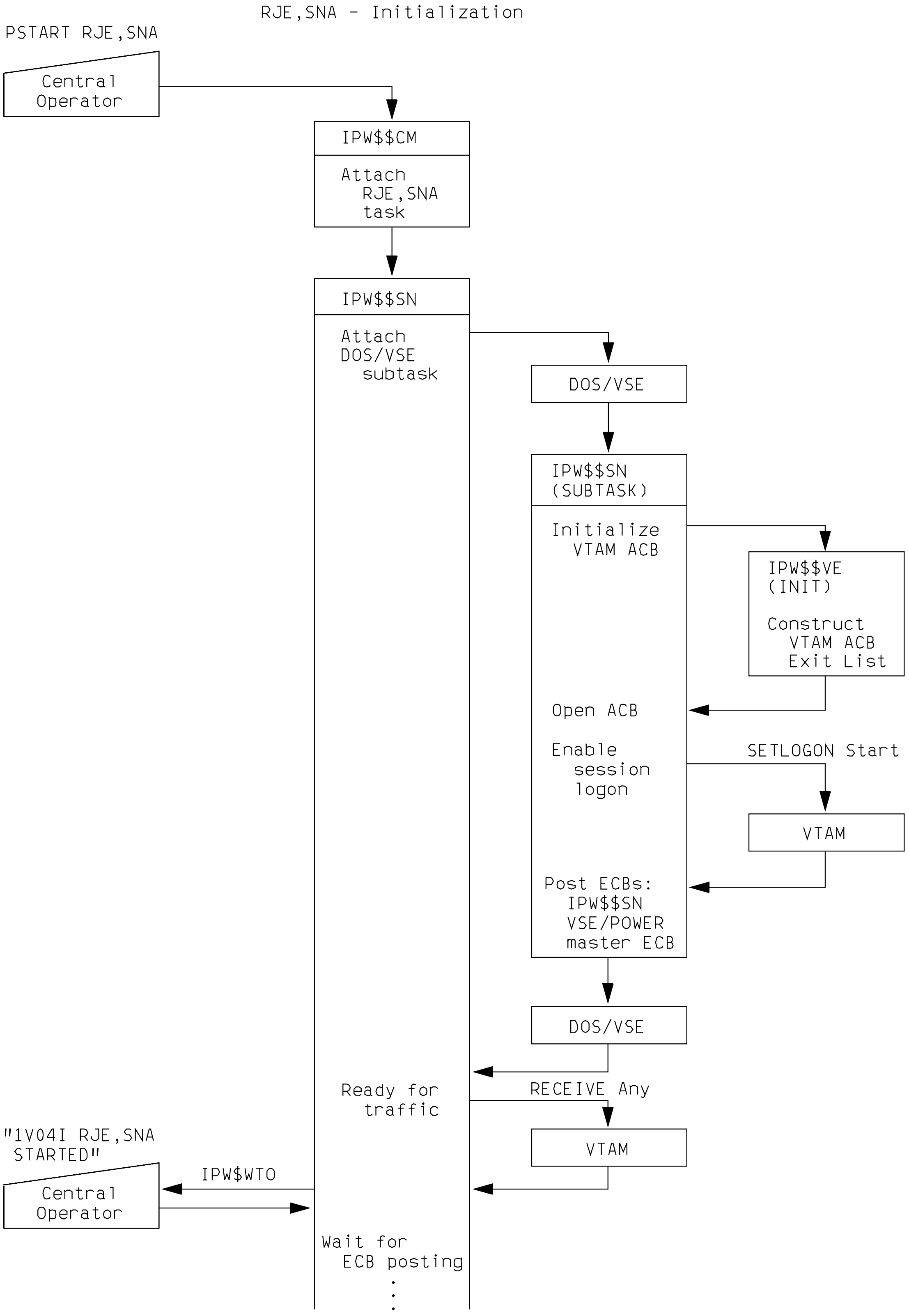


Figure 111 (Part 1 of 9). RJE,SNA Execution Flow

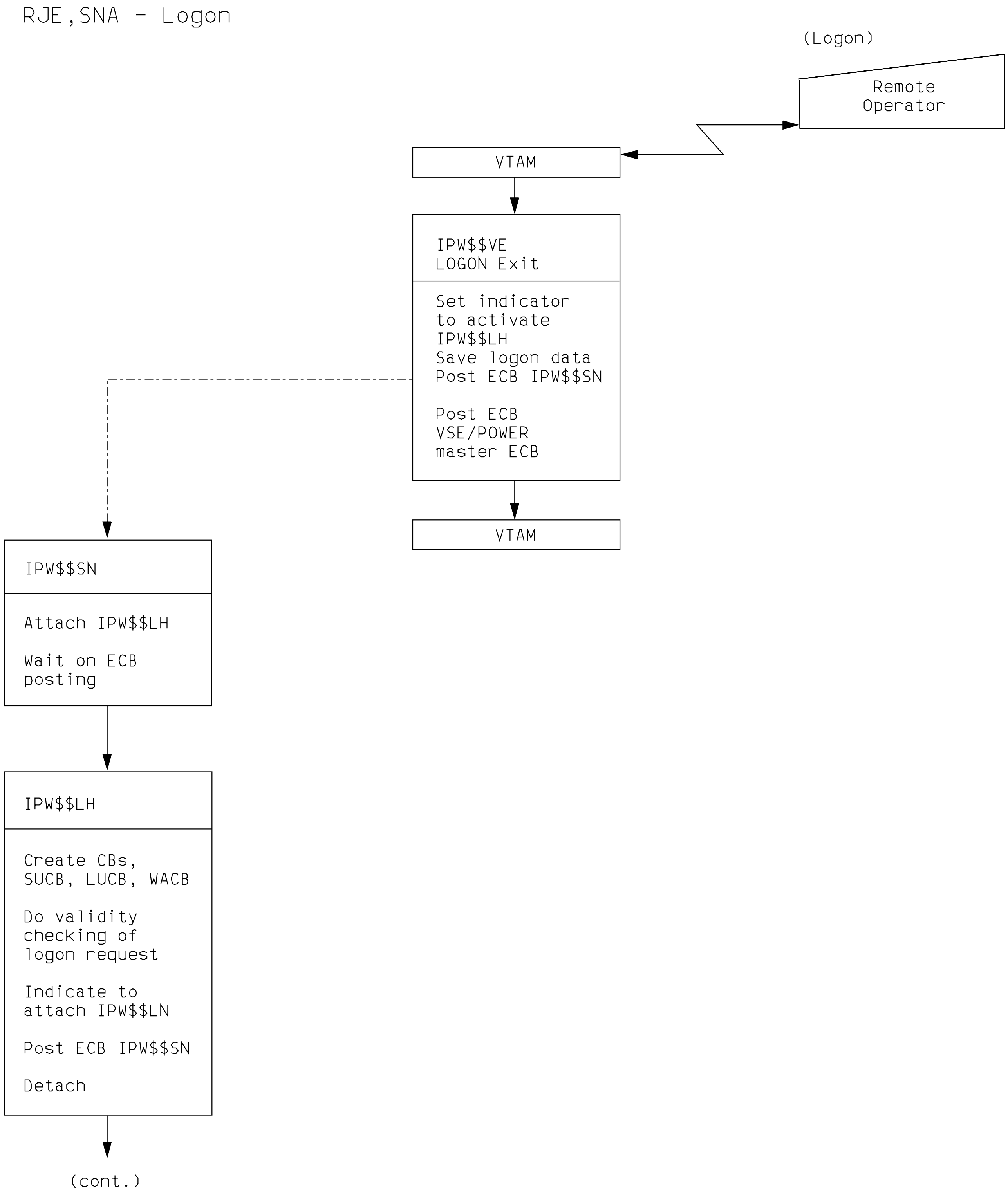


Figure 111 (Part 2 of 9). RJE,SNA Execution Flow

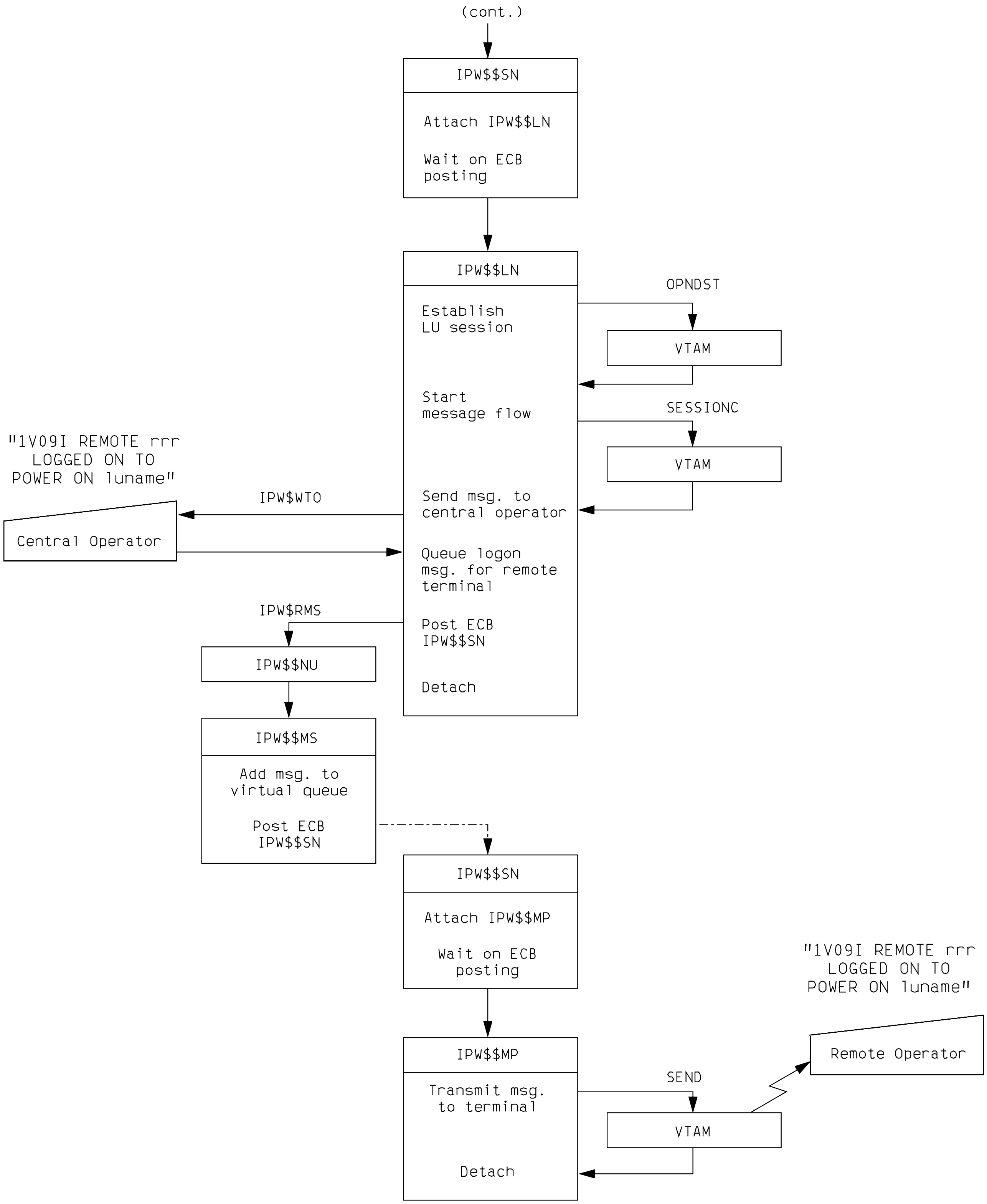


Figure 111 (Part 3 of 9). RJE,SNA Execution Flow

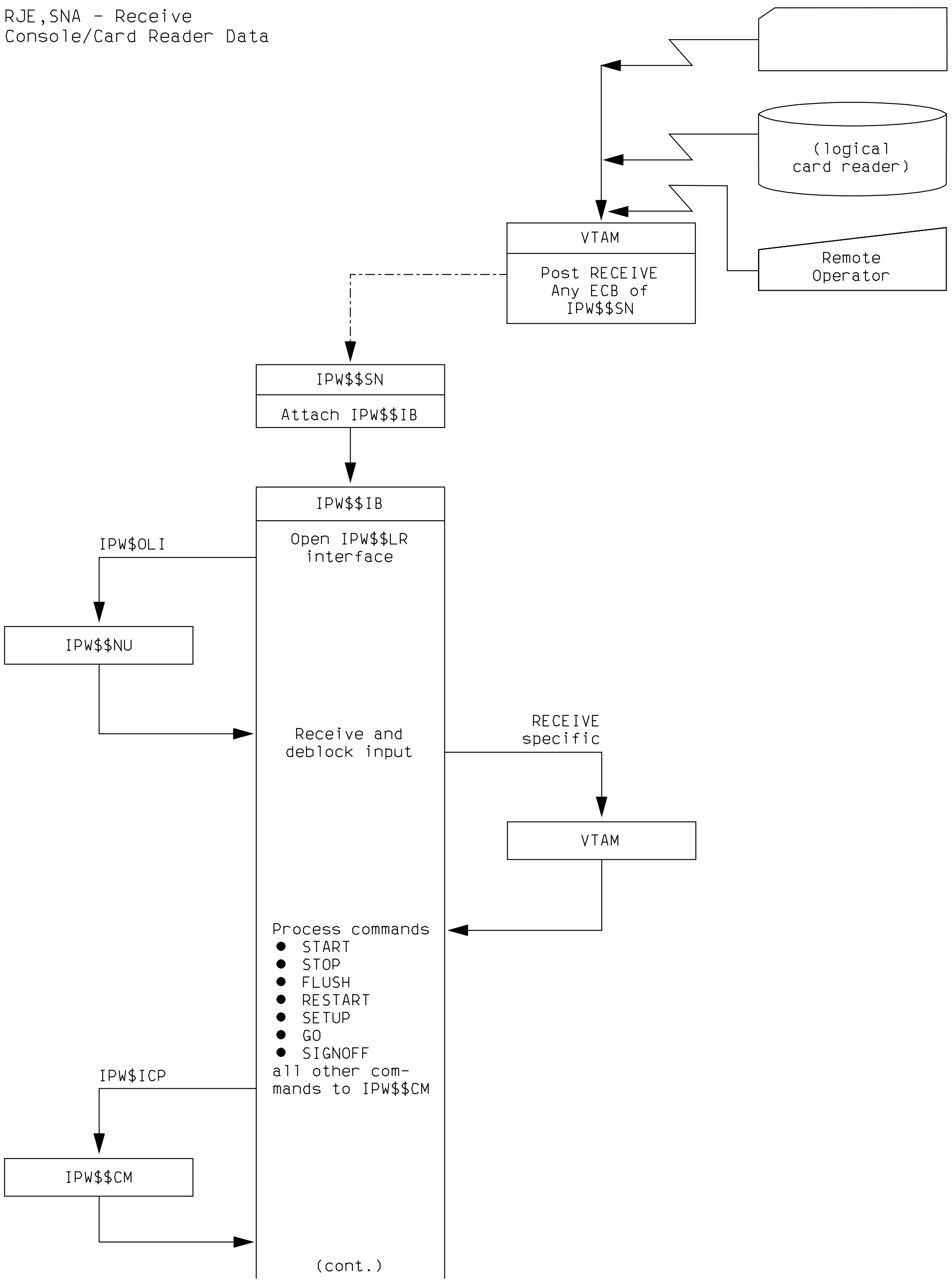


Figure 111 (Part 4 of 9). RJE,SNA Execution Flow

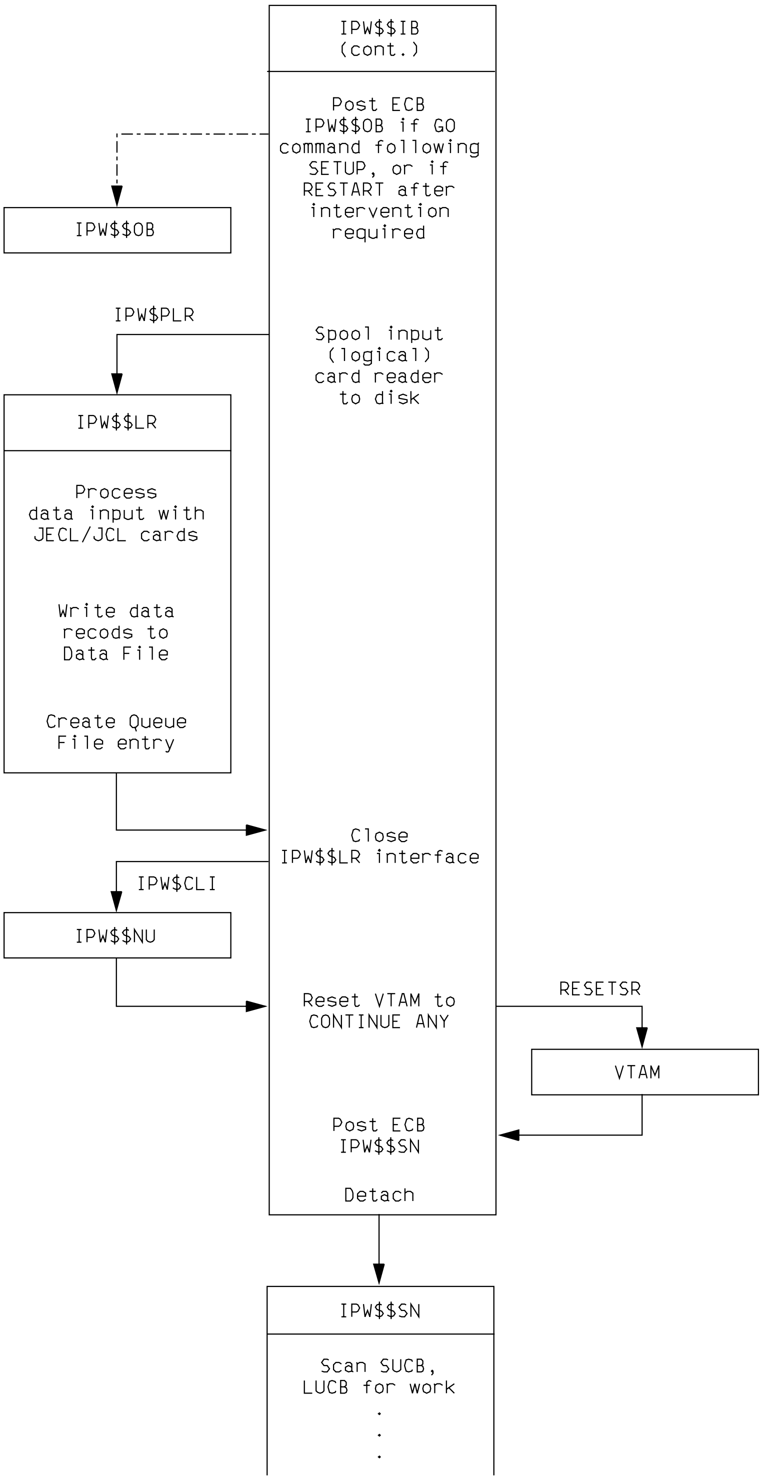


Figure 111 (Part 5 of 9). RJE,SNA Execution Flow

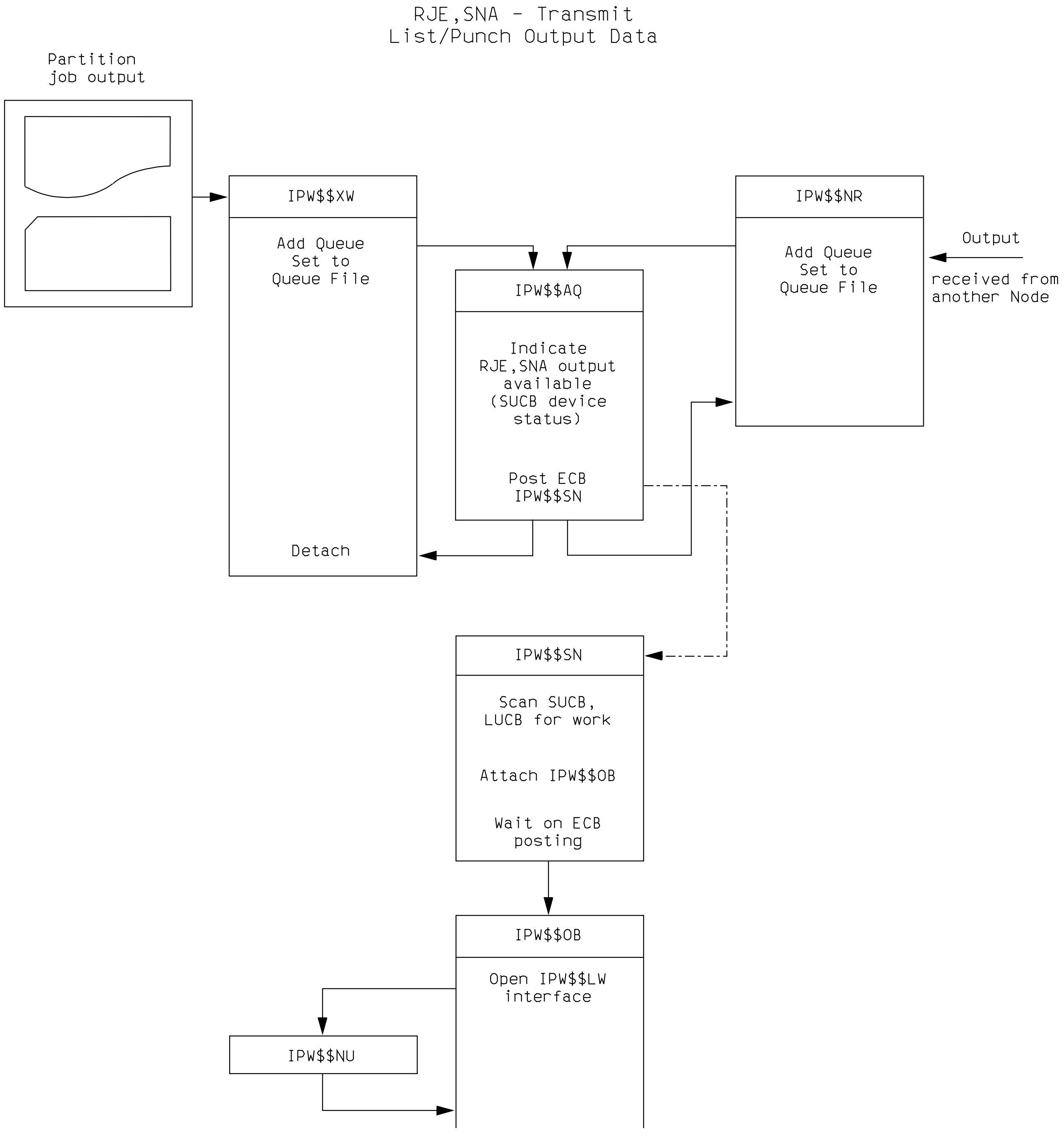


Figure 111 (Part 6 of 9). RJE,SNA Execution Flow

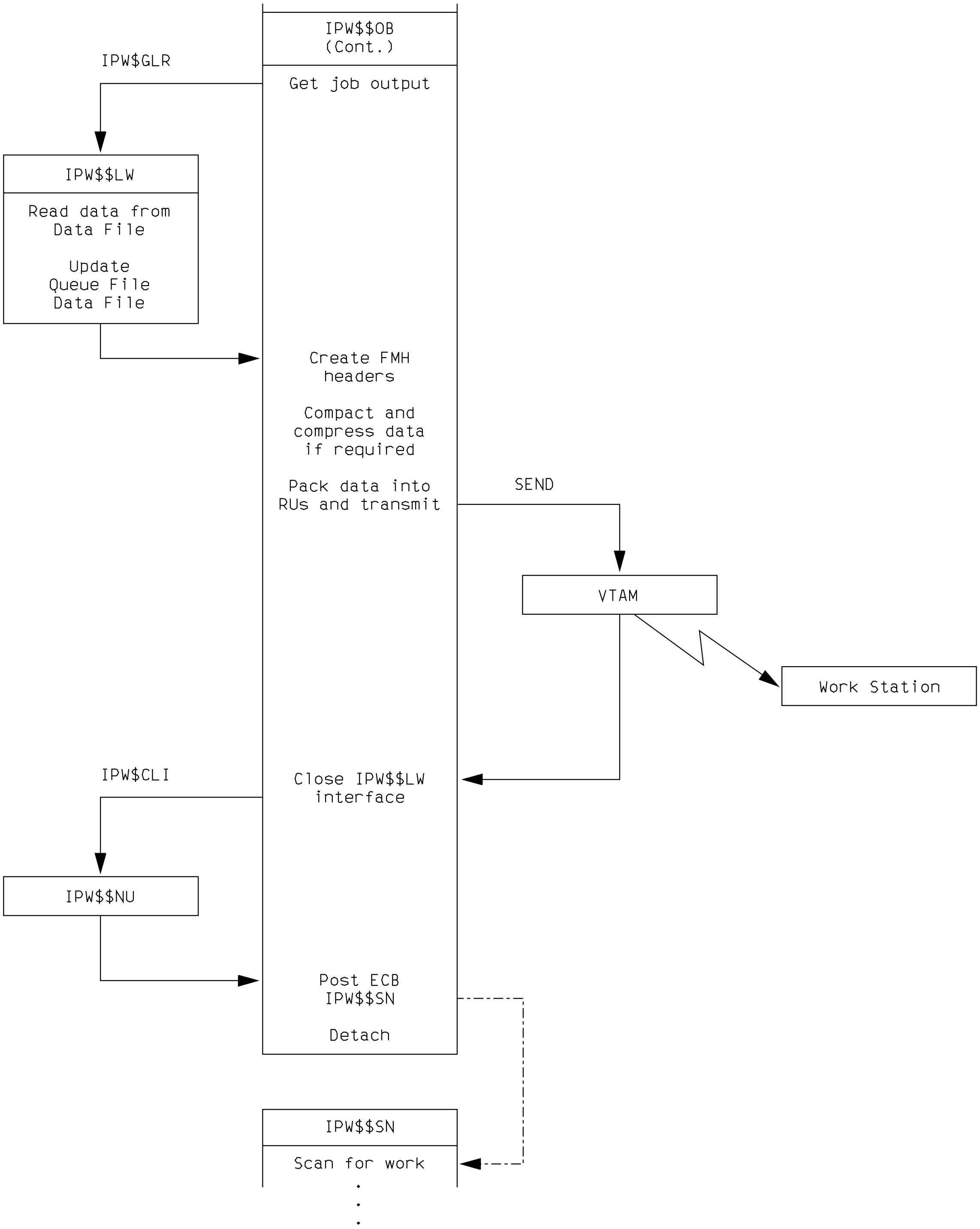


Figure 111 (Part 7 of 9). RJE,SNA Execution Flow

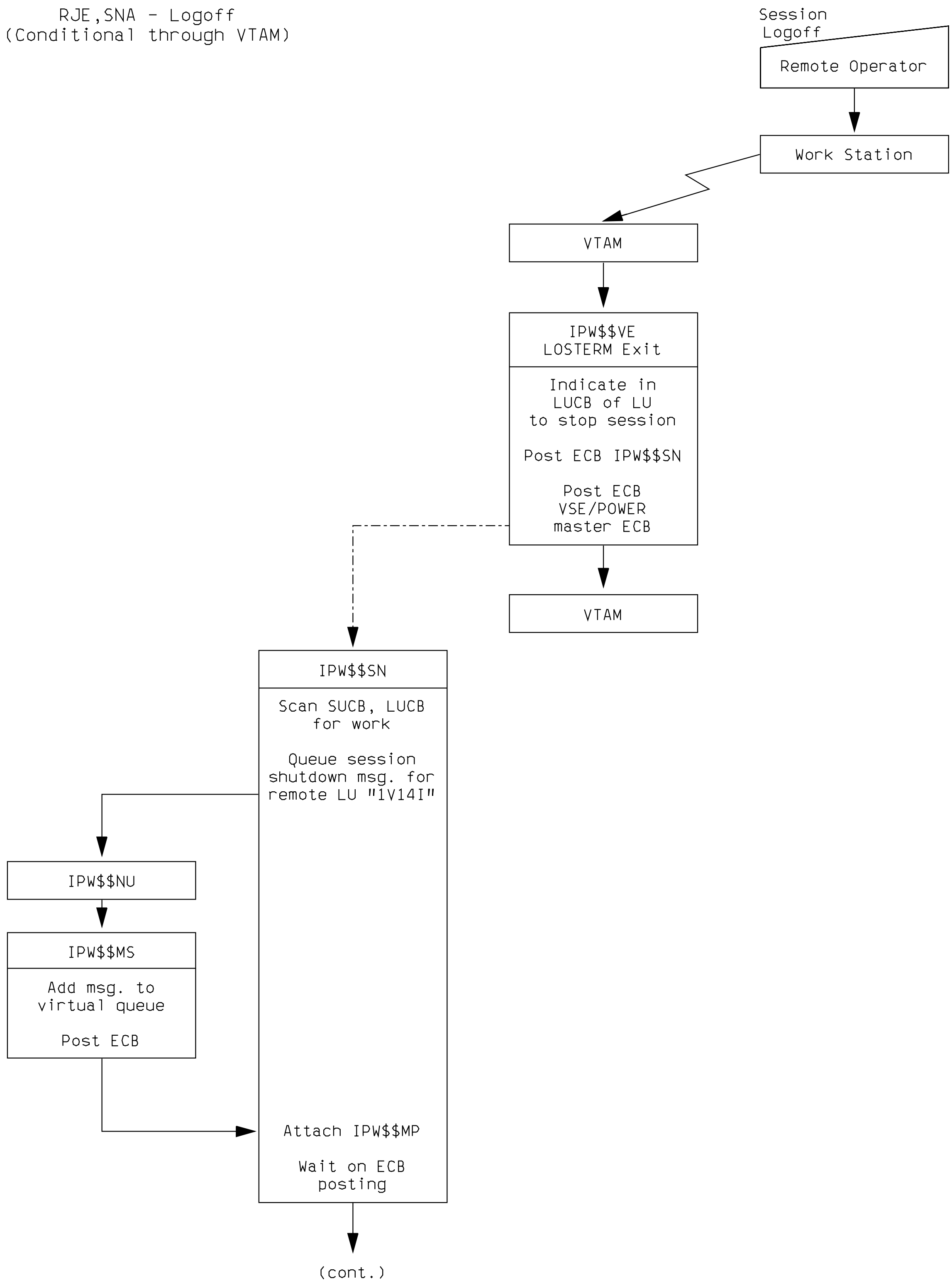


Figure 111 (Part 8 of 9). RJE,SNA Execution Flow

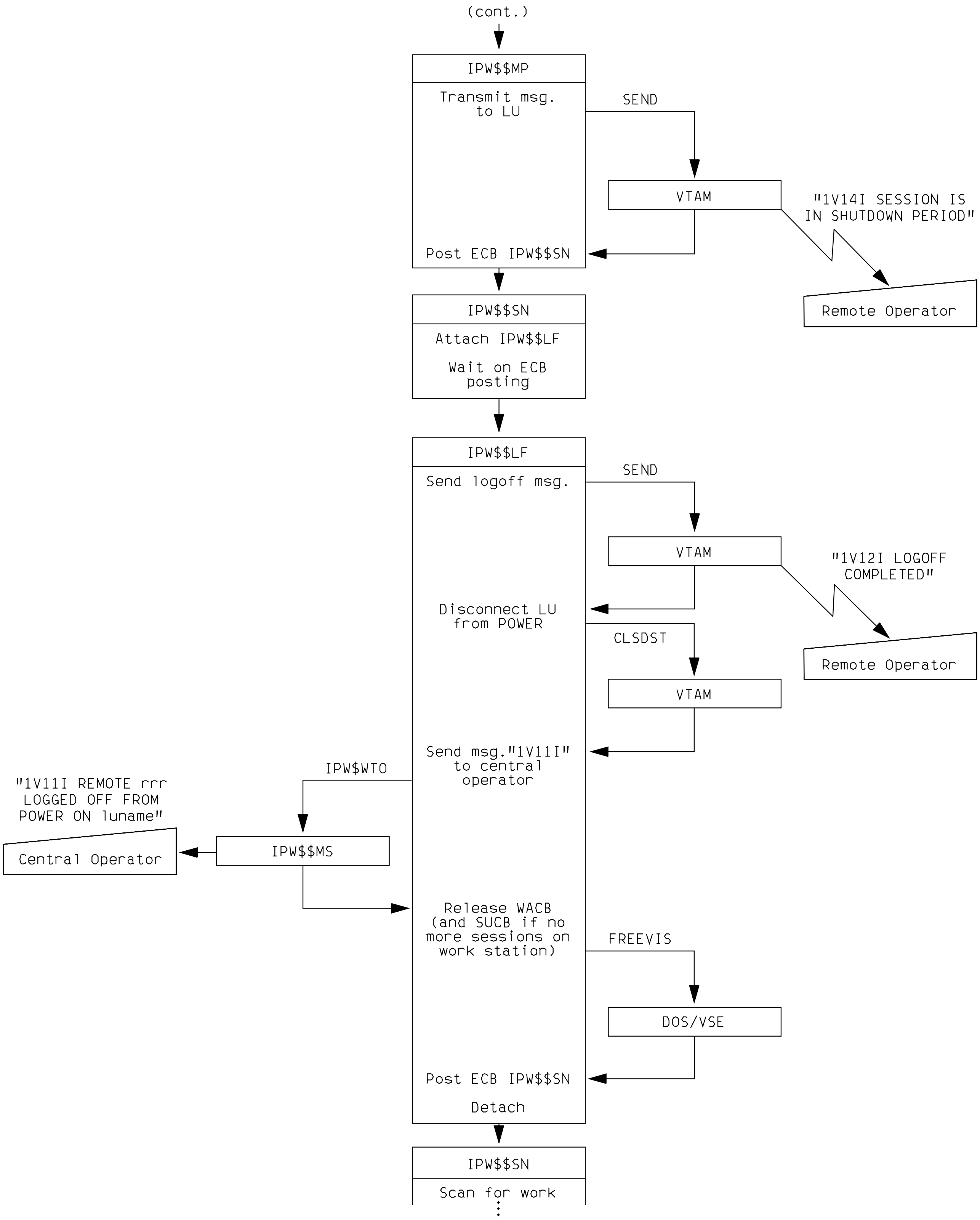


Figure 111 (Part 9 of 9). RJE,SNA Execution Flow

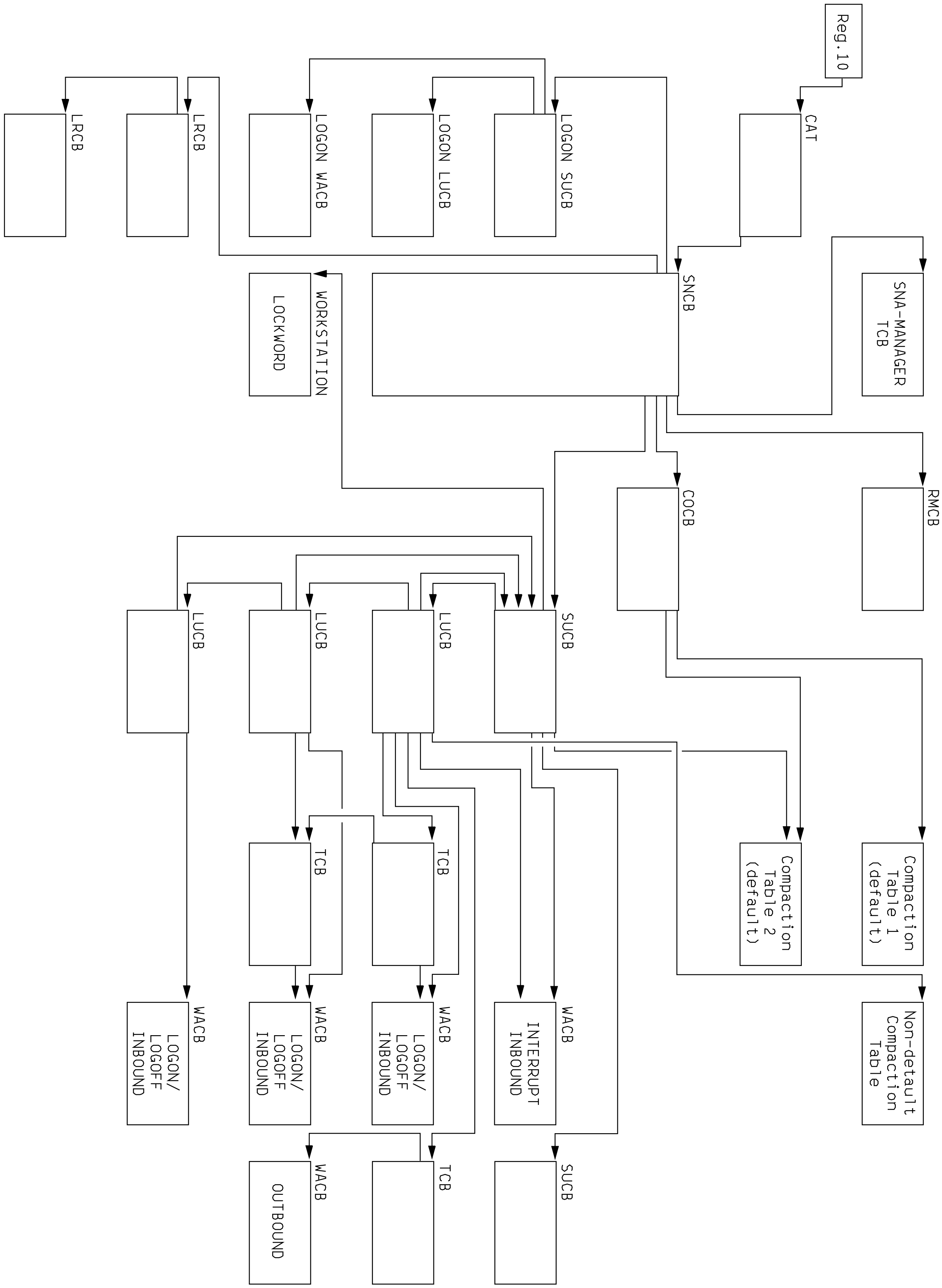

Reg.10
CAT
SNCB
SNA-MANAGER TCB
LOGON SUCB
LOGON LUCB
LOGON WACB
LRCB
LRCB
WORKSTATION LOCKWORD
COCB
RMCB
SUCB
LUCB
LUCB
LUCB
Compaction Table 1 (default)
Compaction Table 2 (default)
TCB
TCB
Non-default Compaction Table
WACB INTERRUPT INBOUND
WACB LOGON/ LOGOFF INBOUND
WACB LOGON/ LOGOFF INBOUND
WACB LOGON/ LOGOFF INBOUND
SUCB
TCB
WACB OUTBOUND

Figure 112. RJE,SNA Control Block and Work Area Chaining

Appendages

Page Fault Appendage

If a page fault occurs, normally the partition is placed in a wait state, until the processing of the page fault is completed.

When a page fault overlap appendage linkage is established, the partition remains dispatchable in order to enable selection of another private task (within the partition) under control of its private multi-tasking routine.

The appendage routine for the VSE/POWER partition is entered from the page manager routine in the supervisor on two conditions:

- The partition sustains a page fault (pre-processor)
- Handling of a page fault is completed (post-processor).

The page fault pre-processor (always entered in Amode-24), will take the following actions:

1. Check if page fault address lies within VSE/POWER partition. If not, return to supervisor with indication that the page fault must be resolved by VSE/AF.
2. Save the task status, address of instruction from PSW, and general registers (taken from the partition save area), in the TCB, because of page handling overlap by the supervisor later on.

   If the task has operated in Amode-31 (currently only valid for MVCL in the Getvis Move Routine) as retrieved from the stored PSW, then preserve the Amode indication in the TCB.

   If the task uses access registers, the access-register mode (retrieved from the PSW stored in the partition save area) and the current values of the access registers are saved.
3. Simulate a IPW$WFP macro instruction (put TCB in P state). This action is transparent to the task management routine.
4. Change the address of the next sequential instruction in the PSW to the entry of VSE/POWER task management, because of page handling overlap by supervisor later on.
5. Queue the page fault request within an internal queue (in TCB chain), unless no page fault is being currently handled for the VSE/POWER partition.
6. If no page fault is currently handled the request is returned directly to the page manager routine in the supervisor. If a page fault is currently handled, a request of zero is returned to the page manager.

The page fault post-processor will take the following actions:

1. Post the task, for which the page fault handling is completed, dispatchable (in TCB reset P state).
2. Post the partition dispatchable, because the partition may be SVC 7 bound if all tasks were waiting.
3. Clear page fault request.
4. Examine the internal page fault queue (in TCB chain).

   If another page fault is found, it is passed to the page manager routine in the supervisor. If no other request is found, a page request of zero is returned to the page manager.

   Note: The page fault currently handled for this partition and the address of the related TCB are saved in the appendage routine itself.

Attention Interface Appendage

Refer to “Initiation of the Permanent Command Processor Task” on page 165.

RJE,BSC and PNET,BSC/CTC Channel End Appendage

During VSE/POWER initialization a modification is made to the PIB of the VSE/POWER partition in order to allow for a channel end appendage used for all RJE,BSC and PNET,BSC/CTC I/O operations. All RJE,BSC and PNET,BSC/CTC CCBs contain the address of the same channel end appendage routine, which is located in the VSE/POWER nucleus.

The appendage routine gets control from the VSE/AF I/O interrupt handler whenever an interrupt is received from a BSC line or channel-to-channel adapter. It then performs the following functions:

- If the line is an RJE line:
  1. It queues the LCB to an LCB chain that will be processed by the line manager.
  2. It posts the line manager ECB, the VSE/POWER master ECB, and sets the VSE/POWER partition dispatchable.
- If the line is a PNET line or CTCA:
  1. It queues the input/output buffer to the buffer queue anchored to the TCB of the PNET driver.
  2. It posts the PNET driver ECB, the VSE/POWER master ECB, and sets the VSE/POWER partition dispatchable.

Control is then returned to the next sequential instruction in the VSE/AF supervisor.

Hot Reader Appendage

The supervisor passes control to this appendage whenever an unsolicited device end interrupt for a unit-record device is recognized.

The reader TCBs are scanned on cuu number to locate the task concerned with the interrupt. If the matching task is inactive, it is posted dispatchable. The VSE/POWER partition is set dispatchable. In all other cases, no action is taken.

SVC 0/3 Appendage

When the SVC 0 supervisor routine determined that the I/O request is for a spool device, this appendage is entered. Control is also passed to this routine to quiesce an I/O request whenever the “Quiesce I/O Supervisor” routine detects that an I/O request for a spooled device is still outstanding.

First the appropriate task list entry in the partition control block is located. If not found, return is made to the supervisor indicating that the I/O request should be handled by the supervisor. If a previous I/O is still being handled, return is made to the supervisor indicating that the I/O request should be queued for re-SVC.

The following actions are taken for an SVC 0:

- The address of the CCB is stored in the task list entry of the partition control block.
- The related execution processor task ECB is posted to let the task simulate the request.
- The VSE/POWER master ECB is posted (in CAT) after selection later on.
- The VSE/POWER partition is set dispatchable.

The following actions are taken for an SVC 3:

- If the quiesce request is not for a device intercepted by an execution reader task, return is made to the supervisor.
- Otherwise the quiesce request is propagated to the execution reader and the execution reader task is posted. The user task is put “VSE/POWER service bound” until the I/O request is completed.
- The VSE/POWER partition is set dispatchable.

Then control is returned to the supervisor with appropriate return code set in register 15.

SVC 90/91 Appendage

The supervisor passes control to this appendage whenever an SVC 90 or SVC 91 interrupt is recognized. The address of the account information is stored in the reader entry of the partition control block. The execution reader task ECB is posted. The VSE/POWER partition is set dispatchable for task selection by the supervisor.

Interval Timer Appendage

The supervisor passes control to this appendage whenever the interval timer expires. The routine sets a flag in the CAT, recording that the timer interval expired and posts the VSE/POWER Master ECB. The VSE/POWER partition is set dispatchable. Return is made to VSE/AF to the instruction that follows the interrupt.

JCL End-of-Job Appendage

This routine is invoked by Job Control at end-of-job time to pass the last and the highest return code of the job in printable format to VSE/POWER. The return codes are passed in registers 0 or 1, respectively, and stored in the partition control block of the partition concerned. The routine returns then to the caller.

Appendage Summary

A summary of the appendages is given in Figure 113.

Event	Appendage	Task Selection Action	Control Blocks
Page fault occurred	Page Fault (pre—processor)	Place current task in wait state, reenter task selection.	TCB
Page fault completed	Page Fault (post—processor)	Make task dispatchable, activate partition.	TCB
Attention interrupt	Attention Interface	Make CP task dispatchable, activate partition.	TCB,CPB
Unsolicited device end	Hot Reader	Set RDR task dispatchable, activate partition.	TCB
BSC channel end	Channel END	Set RJE,BSC line manager (LM)/ PNET driver (LD) dispatchable, activate VSE/POWER partition	TCB,LCB NCB
SVC 0 inter—cepted	SVC 0	Set XR/XW task dispatchable, activate partition.	TCB,PDB
SVC 3	SVC 3	Set XR task dispatchable, activate partition.	TCB
SVC 90/SVC 91 interrupt	SVC 90/SVC 91	Set XR task dispatchable, activate partition.	TCB,PDB
Expiry of interval timer	Interval Timer	Set VSE/POWER partition dispatchable	CAT
End—of job step	JCL End—of—Job	Save highest return code in partition control block	PDB

Figure 113. Appendage Summary

VSE/POWER Shared Spooling Function

The VSE/POWER Shared Spooling function causes a new VSE/POWER task, called the timer task, to be attached. The timer task attaches a VSE/AF subtask.

The timer task provides a time-slicing mechanism for the sharing of the VSE/POWER queue file, data file, and account file among several VSE/AF systems. The VSE/AF subtask receives timer interrupts from VSE/AF.

When a VSE/POWER system issues a LOCK request for access to the shared files mentioned above, a time interval is set when the LOCK request is honored. The timer task regains control when the time interval expires and issues an UNLOCK request for the shared files. To prevent any updating of the queue file without it being locked, the timer task locks the DMB until a new LOCK request is completed successfully.

Timer Task

The timer task maintains time intervals with exclusive write access to one shared queue file and data file, and, if applicable, one shared account file. The shared spooling environment is controlled by four time intervals which are referred to as T1, T2, T3, and T4.

```
A          B    C            A
|--- T1 ---| T2 |---- T3 ----|
```

One specific processor may have successfully obtained a lock on the queue file (point A in the diagram). At point A, a time interval of T1 seconds is set by the timer task. T1 is the maximum time this processor has exclusive write access to the queue file and the data file. When T1 has expired (point B), or earlier if there was no work to do, the timer task issues an unlock for the queue file.

Time interval T2 is then set up as a means to control the operation of processors of different internal speeds. During this time interval, VSE/POWER tasks run normally except for the fact that the timer task has the DMB reserved. When time interval T2 has elapsed (point C), the timer task checks to see if any task has made a reserve request for the DMB. If so, the timer task issues a lock request for the queue file and enters the cycle again (point A). If no request is outstanding, the timer task sets up time interval T3 and enters a wait for its expiration or for a task to issue a reserve request for the DMB. At this point the timer task issues a lock and reenters the cycle (point A) to ensure that nothing has entered the queue that should be processed.

The fourth time interval, T4, is used as a control to check for system abnormalities whenever a lock request is sent to VSE/AF. It can happen that the lock request is not fulfilled; there are several reasons for this:

- Another system could be recovering and thus has the queue file locked for a period of minutes.
- Another system could be saving the account file, which must also complete before control can be released.
- Another system could be performing a POFFLOAD BACKUP function.
- Another system could be performing a PDISPLAY VIO command.
- Another system could not re-build the queue file after a write I/O error, issued message 1QF7A and continued with the storage copy of the queue file, thereby not giving up the queue file lock.
- Another processor that has locked the queue file may have a job that is both in a higher priority partition than VSE/POWER and in a loop; thus it cannot issue an UNLOCK.

If one of these cases occurs and the time interval T4 expires, a warning message (1QB6I) will be sent to the operator and VSE/POWER will set up time interval T4 again.

The timer task performs its time slicing and LOCK/UNLOCK within a VSE/AF subtask. This ensures that the VSE/POWER main task and its internal subtasks do not enter wait state.

After the timer task has obtained the queue file in update mode it examines the 'queue owning sysid' field. If the previous system abnormally terminated all queue record blocks are read in by means of the IPW$RDQ macro instruction and queue file recovery is invoked via IPW$IRY REQ=QUEUE macro. All queue record blocks are written back to disk after recovery is performed; the appropriate refresh bits are set to indicate other shared systems, that the queue record blocks in storage are obsolete.

Otherwise, the timer task routine examines each entry in the refresh table if the appropriate queue record block must be refreshed and if so, the queue record block is read, moved into VIO space via the IPW$RDQ macro instruction and the refresh flag is turned off. This operation will be performed in a loop, one queue record block at a time, until the entire refresh table is processed.

If a PRESET command was given by the system operator, queue file recovery is invoked by means of the IPW$IRY REQ=QUEUE macro instruction in order to perform partial recovery for the system(s) specified in the command.

When the timer interval (T1) ends, the timer task will make its final update to the queue file and then release the 'queue file' lock. All the write operations performed by the timer task use the same general algorithm. As for the read, each entry in the refresh table, representing a queue record block, is examined for any change by the own processor indicated by the “modified” flags. If so, the appropriate queue record block is written back to disk by means of the IPW$WTQ macro instruction and the modified flag is reset, while the refresh flag will be set for all other systems.

If PNET is used in conjunction with shared spooling, then an additional table, called node attached table (NAT), is initialized in the DMB. This table is used to communicate between sharing systems those adjacent nodes that are currently attached for networking. Whenever a node is signed on an entry is made in the NAT. When the node is terminated then the entry is removed.

Whenever a job or output is destined for a node which is not directly attached to this sharing system, the add-to-queue function checks the NAT for the required node name. If an entry is found then the job or output available bit is set to inform any other sharing systems that something has been placed in the transmission queue.

The NAT is part of the master record and thus is written to disk and read from disk by the timer task (IPW$$TI) during every time slice.

There is also another table in the DMB, called the 'Shared Remote-Id Table' (SRT), which is used in a similar way to the node attached table. Every time output becomes available for a remote work station which is not attached to the system producing the output, a bit is set on for the corresponding remote-id in the SRT. This informs the other sharing systems that there is output in the queue for them. This table is also controlled by the timer task (IPW$$TI).

If the first element of the 'wait for run' subqueue has changed, the TES task is posted. That the 'wait for run' subqueue has changed is indicated by a bit in the DMB and is set on for all CPU ids (by IPW$$TQ), whenever a system updates the address of the first element of the 'wait for run' subqueue.

Queue Control Area (QCA)

VSE/POWER maintains an additional area, called Queue Control Area (QCA), to carry information across shared spool to other systems participating in the shared spooling environment. The QCA contains information about each external device which is waiting for work and messages which are destined for one of the systems sharing the same queue/data file.

In addition the QCA contains information about checkpoints taken for queue entries. Whenever a checkpoint with extended checkpoint information has been recorded for a queue entry (using the spool-access support), this extended checkpoint information is written to the QCA. Therefore the QCA may exist also on a non-shared VSE/POWER system.

The QCA is part of the data file and consists of one or more DBLK groups. The area is taken, when necessary, from the free DBLK group subchains and returned to them when no longer needed. Each DBLK group is divided into DBLKs, whereby each DBLK contains an integer number of compartments, called slots. In front of each DBLK is a DBLK-header, which contains information about the number of used slots within the DBLK as well as free space information. The slots have variable length. The following slot types are supported:

- Waiting for work (WFW) slot
- Nodal message record (NMR) slot.
- Checkpoint (CKP) slot.

Each slot consists of a slot header, also referred to as prefix, and the slot body. The slot header will be identical for all slot types, while the slot body differs from one slot type to the other. The slot header is four bytes long and contains information about the slot owning system (SYSID), slot type and the slot body length. The slot body of the waiting for work slot, contains among others the four classes and the logical destination names the external device is supposed to process. The slot body of a nodal message record slot is the NMR itself.

The slot body of a checkpoint slot contains among other information the checkpoint response control record. Whereas a WFW or NMR slot is owned by one specific system, the CKP slot is owned by a queue entry, which might be available for processing by each of the various systems. An 'artificial' owning system of X'0A' is defined for a CKP slot, in order to keep the code changes for CKP slots as small as possible (mainly because a DBLK of the QCA is freed, if the numbers of slots/SYSID within the DBLK-header is zero). Figure 114 on page 338 illustrates the slot DBLK structure.

The slot DBLKs are double-threaded. Each slot DBLK points to the next slot DBLK as well as to the previous slot DBLK by means of relative DBLK numbers. The address of the first slot DBLK is contained in the master record.

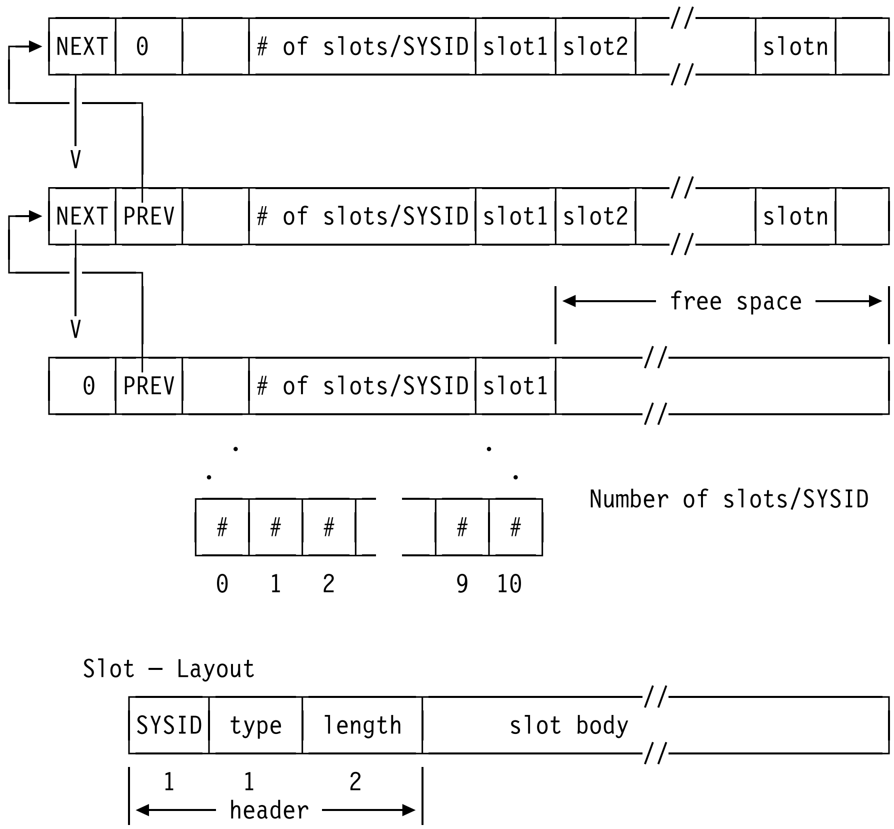


Figure 114. Slot-DBLK Structure

The following function routines make up the slot manager, which is part of the IPW$$SQ module.

- Build slot routine
- Delete slot routine
- Post slot routine
- Process slot routine
- Clear slot routine
- Read slot routine
- Free entire Queue Control Area (QCA)

Build Slot Routine: The build slot routine is entered in response to a IPW$IQS REQ=BUILDSLOT macro instruction issued by a VSE/POWER task. The macro expansion has loaded the slot type identifier in register 0. The routine builds one of the following slots:

1. 'waiting for work' (WFW) slot
2. 'nodal message record' (NMR) slot
3. 'checkpoint' (CKP) slot.

On entry, the routine acquires virtual storage for the slot DBLK area, if not already present. The slot DBLKs are then scanned to find a slot DBLK with enough free space to accommodate the new slot to be built. If found, the slot, consisting of header and body, is created and the slot DBLK is written back to disk.

Otherwise a new slot DBLK is acquired and chained as last entry in the slot DBLK chain. If no QCA exists or the current allocated DBLK group is exhausted, an attempt is made to allocate a new DBLK group from the data file. If successfully obtained, the first DBLK of the just allocated DBLK group is initialized and the slot is constructed in this slot DBLK. The slot DBLK is then written back to disk and chained as last element of the slot DBLK chain.

If, however, no free DBLK group is available, the operator is informed about the spool space shortage via message 1Q38A and the task is placed in wait state until spool file space becomes available.

The build slot routine is invoked in following cases:

1. Whenever the get next queue entry function routine (IPW$$NQ) finds no queue entry eligible for processing by a device service task.
2. Whenever the message distributor routine (IPW$$MX) detects that the nodal message record to be forwarded is destined for another system sharing the same queue/date file.
3. When the message handler (IPW$$MS) detects that the nodal message record to be forwarded is destined for remote node which is attached to another system sharing the same queue/data file.
4. When the spool-access support task (IPW$$XTG) receives a checkpoint control record with extended information and passes this request to the logical writer (IPW$$LW) which issues the macro IPW$IQS. If already a checkpoint slot exists for this queue entry, the already existing checkpoint slot is searched. If the new checkpoint slot fits into the same DBLK containing the old checkpoint slot, the new checkpoint slot is written into this DBLK. If the new checkpoint slot does not fit into the same DBLK containing the old checkpoint slot, the old checkpoint slot is deleted not before the new checkpoint slot has been successfully written into another DLBK.

Delete Slot Routine: The delete slot routine is entered in response to a IPW$IQS REQ=DELSLOT macro instruction issued by a VSE/POWER task. The routine deletes one of the following slots:

1. A 'waiting for work' slot, owned by the local system and matching device name. The device name is contained in the external device control block (EDCB) pointed to by register 1.
2. A 'checkpoint' slot identified by the queue record addressed by TCBQV and the queue record number contained in TCBQW.

On entry, the routine acquires virtual storage for the slot DBLK area, if not already present. The slot DBLKs are then scanned to find the slot in question. If found, the slot is removed from the slot DBLK and the slot DBLK header and the master record are updated; The number of WFW-slots in the master record is decremented by one, if the WFW-slot was not yet posted and the number of active slots/SYSID is decremented.

The delete slot routine is invoked in the following cases:

1. Whenever the get next queue entry function routine (IPW$$NQ) finds a queue entry eligible for processing by the calling device service task and the task was waiting for work.
2. Whenever the device service task is forced to stop immediately, but was waiting for work; in other words a 'waiting for work' slot was built for the task.
3. When the spool-access support task (IPW$$XTC) receives a 'delete checkpoint information' request and passes this request to the command processing routine (IPW$$CL) which issues the macro IPW$IQS. If a checkpoint slot exists for this queue entry, the existing checkpoint slot is deleted as well as the checkpoint information (record number and copy number) within the queue record.
4. When the spool-access support task (IPW$$XTG) receives a checkpoint control record without extended information and passes this request to the logical writer (IPW$$LW). If already a checkpoint slot exists for this queue entry, the logical writer (IPW$$LW) issues the macro IPW$IQS to delete the existing checkpoint slot. The new checkpoint information (record number and copy number) is just recorded within the queue record.

Post Slot Routine: The post slot routine is entered in response to a IPW$IQS REQ=POSTSLOT macro instruction issued by a VSE/POWER task whenever the add queue entry routine (IPW$$AQ) adds a queue entry in the LST/PUN queue where a 'to=' userid is specified. On entry, the routine acquires virtual storage for the slot DBLK area, if not already present. The slot DBLKs are then scanned for a 'waiting for work' slot which is not owned by the local system and the 'to' userid of the just added queue entry matches one of the logical destinations defined in the slot body of the WFW slot. If such a WFW slot is found, the output available flag is set, the number of WFW slots in the master record is decremented and the number of active slots/SYSID is incremented. The following conditions must be fulfilled before the waiting for work slot is posted:

- The slot itself must be unposted (no output available flag set)
- The 'to' userid in the queue record must match one of the logical destinations defined in the slot.
- The class in the queue record must match one of the classes defined in the slot.
- The queue record id must match the queue type defined in the slot.
- The target system id, if one is specified in the queue record, must match the SYSID of the WFW slot owning system.

Process Slot Routine: The process slot routine is entered in response to a IPW$IQS REQ=PROCSLOT macro instruction issued by the timer task at the beginning of the T1 time interval. On entry, the routine acquires virtual storage for the slot DBLK area, if not already present. The slot DBLK chain is then scanned for posted 'waiting for work' slots, owned by the local system, or message slots destined for the local system. In both cases the slot is processed and then removed from the slot DBLK.

The routine performs the following functions:

- For a NMR slot, the nodal message record is passed to the message distributor for further forwarding by means of the IPW$GMS TYPE=DIST macro instruction.
- For a WFW slot, the EDCB chain is scanned to locate the EDCB associated with the external device. An output arrived signal record is then built, added at the tail of the corresponding order queue, if not already done once, and the associated device service task is posted to forward the signal to the DDS.

Such processed slots are then removed from the slot DBLKs. If the last slot DBLK becomes empty, the DBLK is dequeued from the chain. DBLK groups which are no longer occupied by removed slots are returned to the free DBLK group chain. If the QCA becomes empty, the QCA is deleted and the storage used for the slot DBLK area is returned to the virtual storage pool.

Clear Slot Routine: The clear slot routine is entered in response to a IPW$IQS REQ=CLEARSLOT macro instruction issued by a VSE/POWER task. The slot DBLK chain is scanned and each slot in turn is examined if it is a WFW slot owned by the system(s) or a NMR slot destined for the system(s) to be cleaned up. If so, the slot is removed from the slot DBLK and the control information in the slot header and master record are updated accordingly.

The routine is invoked in following cases:

- At VSE/POWER initialization time whenever abnormal warm start is being performed.
- By the timer task at the beginning of the T1 time interval when the master record indicated that the system which had the queue/data file exclusively, abnormally terminated.
- By the command processor task as result of a PRESET QCA command.

Read Slot Routine: The read slot routine is entered in response to a IPW$IQS REQ=READSLOT macro instruction issued by a VSE/POWER task. The routine reads so far just a checkpoint slot identified by the queue record addressed by TCBQV and the queue record number contained in TCBQW. On entry, the routine acquires virtual storage for the slot DBLK area, if not already present. The slot DBLKs are then scanned to find the slot in question. If found, the slot is passed in a virtual storage area allocated by the slot manager. The address of the storage area is contained in TCBXCKPA. The checkpoint remains in the QCA. During this process the DMB is reserved and released. The calling routine has to release the passed storage area, which contains the checkpoint slot, and the address of which is contained in TCBXCKPA.

The read slot routine is invoked in the following cases:

1. When the spool-access support task (IPW$$XTG) receives a a GET request for a queue entry for which a checkpoint slot exists (QRECCKI in the queue record). IPW$$XTG issues the the macro IPW$IQS during the 'open process' of the queue entry.
2. When the spool-access support task receives a request 'retrieve extended checkpoint information' (during GET processing only). IPW$$XTG issues the macro IPW$IQS to get the checkpoint information.

Free QCA Routine: The free QCA routine is entered in response to a IPW$IQS REQ=FREEQCA macro instruction issued by a VSE/POWER task whenever an I/O occurred while accessing the queue control area. The queue control area is deleted, but the DBLK group constituting the QCA are not returned to the free data file space. The virtual storage used for slot DBLK area is returned to the virtual storage pool.

Command/Message Passing Between Sharing Systems

Systems sharing common VSE/POWER queue/data file must by definition appear as one networking node, since it is the common queue file which actually represents the node. Hence central operators on the different SYSID's cannot use networking services to exchange commands or messages. They may display and manipulate the entries of the commonly shared queue file, but they cannot influence any task operating on another sharing system.

Command Passing (PXMIT): The PXMIT command allows to specify a target SYSID to where the enclosed command is to be passed via the QCA for execution. The command format looks as follows:

```
PXMIT SYSID=n,power-command
```

SYSID=n For n specify the identification (1-9) of a sharing VSE/POWER system to where the command should be delivered for execution.
The command is rejected for the own SYSID and is rejected in general on non-shared running systems.

power-command Specify any VSE/POWER command as you would enter it from the central operator console. Find allowed/disallowed commands in the 'NET' column of the "Authorization Table for the Central Op. of a Remote Node" in the *VSE/POWER Networking* manual. Any therein mentioned "(1)" job access limitation is not effective for VSE/POWER commands passed via the QCA using the 'PXMIT SYSID=...' command.

```
Example 1: assuming SYSID=1 central operator wants to trigger a
           PDISPLAY of the VSE/POWER queues on the central operator
           console of SYSID=3, then he would enter:

                   X SYSID=3,D ALL

Example 2: assuming SYSID=1 central operator wants to flush
           job RUNXXX (with jobnumber 17) just executing
           on SYSID=3, then he would use the PCANCEL command
           and enter:

                   X SYSID=3,C RUNXXX,17

           resulting in following messages on the central operator
           console of SYSID=3:
              1R59I FOR SYSID=1 , EXECUTING COMMAND: C RUNXXX,17
              0V16I JOB RUNXXX CANCELED. REQUEST FROM VSE/POWER.
              1S78I JOB TERMINATED DUE TO OPERATOR CANCEL.
```

This version of the PXMIT command may be submitted by the central operator and by RJE users as well as by X-partition users (CTLSPOOL an SAS-CTL) without any restriction.
Independent on the type of submitter, commands will be accepted on the other SYSID according to the 'NET' column of the "Authorization Table for the Central Op. of a Remote Node" in the *VSE/POWER Networking* manual. Any command received via the QCA will be stated on the central op. console by message 1R59I (see example 2).
Any console display lines resulting from execution of an accepted command do not travel back to the command originator (as with PNET), but they appear on the central operator console of the executing system.

Any command rejected for execution (as e.g. the PEND command) is stated by message 1RA7I (this message too does not travel back to the command originator):

```
1RA7I PEND COMMAND NOT ALLOWED ON NODE POWSHR
```

A received command on any SYSID from e.g. SYSID=1 will identify the originating SYSID (thereby overwriting the originating node name, which is the same for all sharing PNET systems) by message:

```
1R59I FOR SYSID=1 , EXECUTING .......
```

When the command originator is an X-partition user with user-id 'XTOOL', message 1R59I will appear as:

```
1R59I FOR SYSID=1(XTOOL), EXECUTING .......
```

Message Passing (PBRDCST): The PBRDCST command allows to specify a target SYSID to where the message is to be passed via the QCA. The command format looks as follows:

```
PBRDCST   nodeid,userid,SYSID=n,'message-text'
```

nodeid For nodeid specify the destination node, if your message is to be sent to another node in the network.
Specify an asterisk (*), if the message is directed to a user at your own node, or if networking is not supported on your system at all.

userid For userid specify 0 or R0 to reach the central operator, specify R1 - R250 to reach an RJE operator, specify another userid according to definition in the *Administration and Operation* to reach an IUI user.

SYSID=n For n specify the identification (1-9) of a sharing VSE/POWER system to where the message should be delivered, when the final 'nodeid' has been reached. SYSID may be seen as a sub-qualification of 'nodeid'.

```
Example 1: assuming both PNET or NON-PNET shared systems;
           you may send a message from SYSID=1
           to central operator of SYSID=3 by

               B *,0,SYSID=3,'MESSAGE FROM SYSID1'

Example 2: assuming PNET link between node POW1 and shared
           node POWSHR with SYSID=1 and SYSID=3 connected
           as follows:
              POW1-----pnet-----POWSHR(1)-Q-POWSHR(3)

           you may send a message from POW1 to central
           operator of SYSID=3 - which has actually
           no PNET link - by:

               B POWSHR,0,SYSID=3,'MESSAGE FROM POW1'
```

The extended version of the broadcast command may be submitted by the central operator and by RJE users as well as by X-partition users (CTLSPOOL an SAS-CTL) without any restriction.

Restrictions: Commands/messages directed to sharing SYSIDs currently not initialized (started) are kept in the QCA and are delivered immediately at VSE/POWER startup. When such preserved commands/messages for e.g. SYSID=5 should be removed from the QCA, then use the following command:

```
                PRESET QCA,5
```

When commands/messages are directed to idling systems, they will not be read out before the t3 interval elapses (default t3=60 sec). Prompt delivery may be enforced with minimal performance impact by specifying e.g. t3=2 sec (see TIME=operand of the POWER macro).

Note: Being in test mode on an idling SYSID awaiting commands/messages to be read out from the QCA, you may trigger write access to the shared queue file by entering e.g. 'PDISPLAY Q'. This command locks the DMB (Master Record), i.e. asks for queue file write access.

This support will not be documented officially, since:
First there is no guarantee for immediate delivery of messages/commands to another sharing system - namely when the recipient idles. Then no VSE/POWER task wants to access the shared queue file in write mode until after t3=60 sec(s) (default) the recipient is forced to look into the shared queue file for any passed message or command.
Secondly, when many messages/commands are passed to another sharing system, it may happen that they are not received in the same sequence as they were entered at the sending system.

VSE/POWER Spool-Access Support Interface

The Spool Access Support interface, referred to in other parts of the manual as SAS, gives a user in a partition other than the VSE/POWER partition the capability to:

- Submit a job to the VSE/POWER RDR queue for later execution in a partition controlled by VSE/POWER or to the XMT queue for transmission to another node in the network.
- Spool list or punch data to the output queues (LST/PUN/XMT).
- Retrieve data from the RDR, LST, or PUN queue.
- Manipulate either the local queues or any queues in another system assuming that the user has sufficient authority.
- Process other VSE/POWER commands, such as the PBRDCST command.
- Process other spool-access-support requests, such as recording, retrieving, or deleting checkpoint information.
- Retrieve job completion messages of jobs submitted via the Spool-access support

The VSE/AF XPCC support is utilized for communication between VSE/POWER and the other partition.

The Spool Access Support function is performed by the following phases:

- IPW$$XM - Master task processing routine
- IPW$$XT - User task mainline routine
- IPW$$XTC - CTL function processing routine
- IPW$$XTG - GET function processing routine
- IPW$$XTP - PUT function processing routine
- IPW$$XTM - GCM function processing routine
- IPW$$XTS - Subroutines
- IPW$$LO - Logical output spooler
- IPW$$PC - Parameter checking routine

Spool Access Support Master Task

The spool access support master task is attached as a VSE/POWER task at VSE/POWER initialization time. No generation option is required for the spool access support.

The task is terminated at VSE/POWER shutdown time, when no other tasks, except the command processor and terminator tasks, exist.

First, the task acquires storage for and initializes the Communicator Information Block (CIB) and the Communicator Information Block 2 (CI2). The CIB and the CI2 are anchored to the CAT and contain the anchor points for the various message queues. If no storage is available, message 1Q08I is issued and the task is terminated. Otherwise, a XPCCB control block is initialized and chained off the CIB. Figure 115 on page 346and Figure 47 on page 134 illustrate the control block relationship. The task then identifies VSE/POWER as user of the XPCC interface by means of the XPCC FUNC=IDENT macro instruction. If an error occurs, message 1QX1I is issued and the task is terminated. The SUBSID macro is used to decide whether the system runs unattended or not. Next, a 'connect any' is done by issuing the XPCC FUNC=CONNECT macro instruction. Again, if an error occurs, message 1QX1I is issued and the task is terminated.

The task then waits for a connection request from an SAS user or for a signal passed by the PEND command processor. The command processor sets termination code 'E' in the TCB and posts the SAS master task when PEND was entered. This causes the task to inform VSE/AF that new connection requests are no longer accepted by means of the XPCC FUNC=TERMQSCE macro instruction.

If a new connection is established (connect ECB of XPCCB posted), storage is acquired for an SAS user task TCB and work area. The SAS user task is then attached with the used XPCCB passed.

If, however, the connection is a 'notify' connection, the VSE/POWER notify task is attached if not already present. If the counterpart of the 'notify' connection is neither VSE/ICCF nor VSE/DSNX, a Communicator Information Element (CIE) is built and chained to the CIE-queue anchored to the CIB. The CIE contains head and tail pointer of the appropriate message queue and status information concerning the 'notify' communication path.

If, however, the connection is a 'heartbeat' connection, the VSE/POWER heartbeat task is attached, if not already present. If already present, this connection is stopped with the appropriate return codes within the XPCCB by issuing the macro XPCC with the function DISCPRG.

A new XPCCB is then built and a new 'connect any' request is established. This process continues until the PEND command is entered.

At VSE/POWER termination time, the SAS master task informs the notify task, if present, to terminate and waits till the notify task is detached. Thereafter the SAS master task informs the heartbeat task, if present, to terminate and waits till the heartbeat task is detached.

Then VSE/AF is informed that the XPCC facilities are no longer used by means of the XPCC FUNC=TERMPRG macro instruction. Before detaching itself, the termination task is posted.

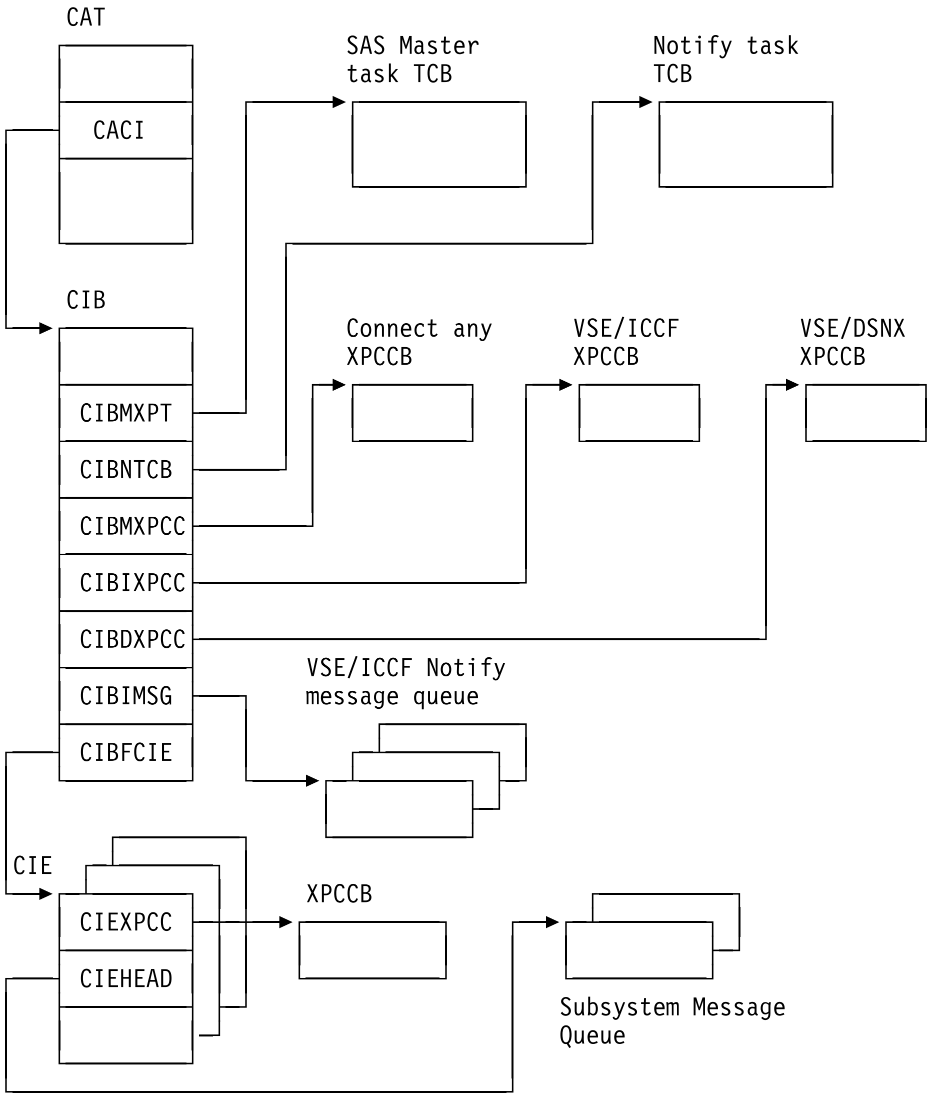


Figure 115. SAS Master Task Control Block Relationship

SAS User Task

The SAS user task is attached by the SAS master master task when a new connection request is honored. The SAS master task has already equipped the task with a work area. The function of the task is to:

- Spool jobs to the VSE/POWER reader or XMIT queue.
- Spool list/punch output to the appropriate queue.
- Retrieve job/output data from the various local VSE/POWER queues.
- Manipulate queue entries in the various VSE/POWER queues.
- Process other VSE/POWER commands and return of results (e.g. PDISPLAY).
- Retrieve job event messages of jobs submitted via Spool-access suppo

After initial house-keeping is done, the task waits for the first data sent from the cross partition user via the new path. VSE/POWER expects that the first buffer received contains a Spool Parameter List (SPL). If this is not the case, the request is rejected and the task then waits for another request.

All messages issued by the SAS user task or on behalf of the SAS user task are collected by message service and passed to the cross partition user.

The communication path is discontinued by the SAS user task either at the end of the function currently performed when the system operator has entered the PEND command (stop code 'E' set in the TCB) or immediately by means of the XPCC FUNC=DISCPRG macro instruction when the operator has entered the appropriate PSTOP SAS command.

It might be possible that the SAS user task encounters a wait condition. Possible wait conditions are:

- No real/virtual storage available to handle request
- No disk space available when spooling job/output data
- Account file full when writing account record
- Resource(s) locked by any other VSE/POWER task.

In all of these situations the SAS user task waits until either disk or storage space becomes available or until the resource is unlocked, unless the "no wait" option was set in the SPL and either a no spool space or account file space condition occurred. In this case, the SAS user task returns to the user indicating the shortage.

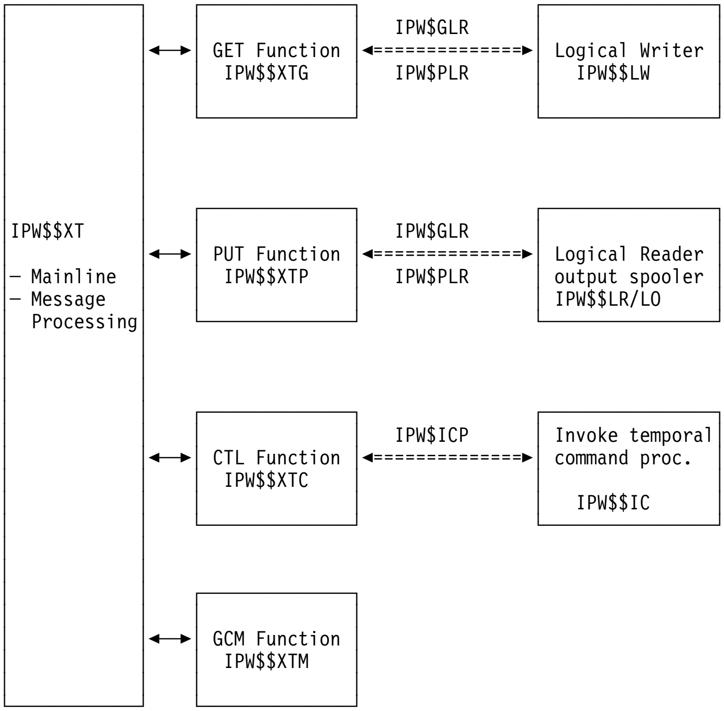


Figure 116. SAS User Task Module Structure

SAS Protocol

The VSE/AF cross partition communication facility (XPCC) is used for data transmission between VSE/POWER and the spool access support user.

The spool parameter list (SPL) interface is used for all general requests and determines which function is to be processed.

The basic method of communication between VSE/POWER and a SAS user consists of exchanging data buffers in both directions. Associated with each buffer are 8-bytes of user data. The user data indicates what type of buffer is sent and which action should be performed by VSE/POWER. The action bytes in the user data define what to do with the passed data buffer. The data buffer may also contain a control record. Only one control record is allowed in a buffer. In general, the SAS user starts the communication by sending a request to VSE/POWER via the XPCC FUNC=SENDR facility. VSE/POWER analyzes the request, performs the necessary actions and passes the result of the request back to the SAS user. The result can be either:

- Return/feedback code, provided by VSE/POWER. The return and feedback codes tell the requestor what happened.
- A data buffer.
- Message(s).

Four types of functions are supported by VSE/POWER:

- GET function
- GET BROWSE function
- PUT function
- CTL function
- GCM function

Each function consists of a set of requests. The possible requests are outlined separately for each function in Figure 117 to Figure 121.

Request: (User->POWER)	Definition
OPEN	Access queue entry, described in SPL (initialize GET function)
DIRECT OPEN	Access queue entry, described in SPL (initialize GET function) by queue record number
OPEN-GENERIC	Access queue entry(ies), according to specified class(es) (initialize GET function). Queue entry must be dispatchable.
GETRQ	Obtain one or more data records.
QUIT	Release accessed queue entry and return to queue with original disposition.
CLOSE	Release accessed queue entry and return queue entry to queue according to VSE/POWER disposition rules ('K' ->'L').
CHECKPOINT	Save current copy number and record number as supplied by user, and extended information, if specified.
RESTART	Restart at a specified record number and copy number.
PURGE	Terminate access to queue entry and delete the entry.
FLUSH-HOLD (Note)	Terminate access to queue entry and put back in queue with disposition 'H' if original disposition was 'D', or with disposition 'L' if original disposition was 'K'.
QUIT and LOCK	Release accessed queue entry and return to queue with temporary disposition 'Y'.
GET OPTB	Get one or entire OPTB structure associated with accessed queue entry.
MODIFY OPTB	Modify one OPTB.

Figure 117. Possible Requests within GET Function

Note: Only applicable for DDS (Device Driving System)

Request: (User->POWER)	Definition
OPEN-BROWSE	Access queue entry, described in SPL (initialize GET function). Queue entry can be in any disposition (D, K, H, L, X, A, Y, *)
DIRECT OPEN-BROWSE	Access queue entry, described in SPL (initialize GET function) by queue record number. Queue entry can be in any disposition (D, K, H, L, X, A, Y, *) and even 'in creation' (SPLGOGIC=ON for DIRECT OPEN-BROWSE for in creation)
GETRQ	Obtain one or more data records.
QUIT	Release accessed queue entry and return to queue with original disposition.
RESTART	Restart at a specified record number and copy number.
GET OPTB	Get one or entire OPTB structure associated with accessed queue entry.

Figure 118. Possible Requests within GET BROWSE Function

Request: (User-&POWER)	Applicable for		Definition
	Job	Out	
OPEN	Y	Y	Initialize PUT function with SPL values
OPEN-APPEND		Y	Initialize PUT function with SPL values for "appendable" Output data.
OPEN-RESTART		Y	Initialize PUT function with SPL values for an Output data set whose spooling was abnormally terminated.
PUTRQ	Y	Y	Spool data to the VSE/POWER Job or Output file.
PUTRQ+SEGMENT		Y	Spool Output data and Segment.
PUTRQ+CHKPT		Y	Spool data, save to the disk file, and reply to user with a unique checkpoint id.
PUTRQ+CLOSE	Y	Y	Spool data + Close the PUT function.
PUTRQ+CLOSE + UPDATE		Y	Spool data + Close the PUT function with updated SPL values.
CHECKPOINT		Y	Save any buffered data to the disk file and reply to user with a unique checkpoint id.
RESTART		Y	Restart at a specified record number.
SEGMENT		Y	Segment any spooled Output data.
SEGMENT+UPDT		Y	Segment with updated SPL values.
CLOSE	Y	Y	Close PUT data function.
CLOSE+APPEND		Y	Close for "Append" Output data.
CLOSE+UPDATE		Y	Close PUT data with updated SPL.
CLOSE+UPDATE + APPEND		Y	Close with update SPL for "Append" Output data.
QUIT	Y	Y	Flush the data set being spooled.
GET-MSG	Y	Y	Obtain the next n message(s).
GET OPTB		Y	Get one or entire OPTB structure of queue entry.
MODIFY OPTB		Y	Modify one OPTB.

Figure 119. Possible Requests within PUT Function

Request: (User-&POWER)	Definition
COMMAND	Pass command to VSE/POWER
GET-MSG	Get next n messages.
QUIT	Flush receiving of queued messages.

Figure 120. Possible Requests within CTL Function

Request: (User-&POWER)	Definition
OPEN-KEEP	Begin the request and get copy of fixed format messages. Retrieved messages may be retrieved again later.
OPEN-DELETE	Begin the request and get fixed format messages. Retrieved messages are deleted by VSE/POWER and cannot be retrieved again.
OPEN-REMOVE	Begin the request and remove all already retrieved messages in the addressed message queue.
OPEN-PURGE	Begin the request and remove all messages contained in the addressed message queue.
MORE	retrieve more messages
REMOVE	remove already retrieved messages.

Figure 121. Possible Requests within GCM Function

CTL Function: If an SPL, requesting a CTL function, is sent by the cross partition user, the SAS user task, calls the SPL parameter checker by means of the IPW$SSJ macro instruction. The parameter checker examines each applicable field contained in the SPL for correctness. If an invalid or missing specification is detected, return is made to the SAS user task with the appropriate return/feedback code set. This code is then passed to the cross partition user and the CTL request is rejected.

If a VSE/POWER command was passed, the invoke command processor routine is called via the IPW$ICP macro instruction and the SAS user task then waits for completion of the command. If, however, a command request is passed in the SPL, the appropriate VSE/POWER command is built and passed to the 'invoke command processor' routine. One of the following commands is built:

- PALTER jobname[,jobnumber[,jobsuffix]][,CCLASS=c],attribute=xxxx
- PCANCEL jobname[,jobnumber]
- PRELEASE jobname[,jobnumber[,jobsuffix]][,CCLASS=c]
- PHOLD jobname[,jobnumber[,jobsuffix]][,CCLASS=c]
- PDELETE jobname[,jobnumber[,jobsuffix]][,CCLASS=c]
- PDISPLAY jobname[,jobnumber][,CCLASS=c]

If messages were queued as a result of the command, the SAS user task builds a message buffer and sends it to the cross-partition user by means of the XPCC FUNC=REPLY macro instruction. VSE/POWER attempts to buffer the messages. The buffer size is the length of the reply area specified by the cross-partition user at SENDR time. If all messages fit into the reply message buffer, the buffer is sent to the cross-partition user with return/feedback code indicating end of message(s).

In the case of a queue display command, VSE/POWER takes a snapshot of the queue(s) and places the result of the snapshot as a LST queue entry in a specific class chain. This queue entry is then accessed by the SAS user task and all messages, describing the current queue contents, are passed to the user without that the user is aware that a LST queue entry is built by VSE/POWER.

In addition, if the passed command is PALTER, PDELETE, PDISPLAY, PHOLD, or PRELEASE, and SPLGO2QN specifies to 'use the direct queue entry number' as passed in SPLXQNUM, the following takes place:

1. Macro IPW$ICP with PASS=QEN is used to hand over the 'direct' option to the desired command processor module.
2. The queue manipulation commands (A|L|H|R) process the request in their corresponding 'direct' routine, from where the 'direct inspect (QRDISPCT)' routine of IPW$$CM is called to determine the eligibility of the queue entry accessed by its number. When not found, that is RC/FB=04/01, then PXPFBKC2 provides detailed information, why not found.
3. The PDISPLAY command uses the direct lookup routine (PSDQLU00) of IPW$$PS, which in turn calls the direct selection routine of IPW$$PS1 by means of IPW$IIS REQ=DIREL. When not found, that is RC/FB=04/0B, then PXPFBKC2 provides detailed information, why not found. When found, the display of only one queue entry is not placed into a snapshot list queue entry, but passed by virtual storage message buffers to save disk I/O time.

If the checkpoint information is to be deleted, the following takes place:

1. The following command is built.
   PDELETE jobname[,jobnumber[,jobsuffix]]
   Note that the operand CCLASS is not used.
2. The IPW$ICP macro instruction is used with the option PASS=DCK to pass the information 'delete checkpoint information' to the command processor.

```
 USER                                 VSE/POWER
   |                                      |
COMMAND:                             wait (RECB)
 send command:                            |
   SENDR (SPL) ------------------->       |
   |                                RECEIVE SPL
 wait (SECB)                          • check SPL
   |                                  • construct command
   |                                  • invoke command processor
   |                                      |
   |                          wait till cmd processor finished
   |                                  • get first message(s)
      <--(user data=EOM,if no more msg)---- REPLY
   |       + message buffer, if any       |
   |                                      |
 if more messages are queued:             |
   |   <.....................................................
   |                                      |                 .
GET_MSG:                               wait (RECB)          .
 request next msg:                        |                 .
   SENDR --(user data=get next msg)---->  |                 .
   |                                      |                 .
 wait (SECB)                            • get next message  .
   |                                    • fill up buffer    .
   |                                      |                 .
     <--(user data=EOM,if no more msg) -- REPLY             .
   |       + message buffer, if any       |                 .
   |                                      |                 .
   |       Loop till end of messages      |                 .
   .........................................................
   |                                      |
   |                                wait (RECB)
   |                                  • wait for next SPL
  ...    wait for any other activity     ....
```

Figure 122. CTL Function Control and Data Flow

GET Function: The SAS user requests obtaining data from the VSE/POWER queues by setting up and sending a Spool Parameter List (SPL). The SPL defines the job name, the job number and job suffix number (optional), class, and queue type of the job/output to be retrieved.

When the SAS user task receives an SPL, it invokes the SPL parameter checker by means of the IPW$SSJ macro instruction to check the SPL for correct specification. If an invalid or missing specification is detected, the SAS user task rejects the GET function and passes back appropriate return and feedback code information.

Otherwise the interface to the logical writer is opened by means of the IPW$OLI macro instruction and the first data record is requested via the IPW$GLR macro instruction. This causes the logical writer to scan the specified queue in order to locate the queue entry addressed by job name, number, suffix and class. If no such queue entry exists or the queue entry is protected (password mismatch), return code with appropriate feedback information are set in the user data field of the XPCCB and a null buffer is sent back. In order to access a queue entry, the appropriate queue entry must be in dispatchable state, which means the disposition must be either 'D' or 'K' and the cross-partition user must be either the originator or owner of the job/output queue entry. Queue entries with a 'to' destination of 'ANY' can be retrieved by any cross-partition user.

If the standard GET-OPEN request specifies also SPLGO2QN to 'use the direct queue entry number' as passed in SPLXQNUM, the following takes place:

1. The communication with the logical writer is opened as for standard GET-OPEN requests, and the logical writer enters the Get Next Queue Set function .
2. IPW$$NQ does not start to search class chains, but reads the queue record by its internal number provided in SPLXQNUM.
3. The direct selection routine 'NQA1' checks, if the retrieved queue record matches the standard GET-OPEN selection criteria as described before. When not eligible, meaning not found, that is RC/FB=04/01, then PXPFBKC2 provides detailed information, why not found. When found, 'direct' and 'non direct' GET access behave the same.

If found, the logical writer passes the job header record and, if present, the data set header record to the SAS user task. These records together with the information extracted from the queue record are used to build an 'extended' SPL, which contains all descriptive information, such as FCB name, UCB name, record format and maximum record length, about the queue entry being obtained. The SAS user task then sends the 'extended' SPL to the cross-partition user by means of the XPCC FUNC=REPLY macro instruction. The cross-partition user may analyze the SPL and take appropriate actions. After the SPL is sent, the SAS user task waits for the next request from the cross-partition user.

If the cross-partition user requests the next record(s), buffer storage is acquired (IPW$RSV) in the length specified as reply buffer size by the cross-partition user or in the max. length supported by VSE/POWER (curr. 64K), whatever is less. The acquired buffer is then filled with data records, which are obtained via the IPW$GLR macro instruction. If the buffer is full or the logical writer has indicated 'end-of-data', the buffer is sent to the cross-partition user by means of the XPCC FUNC=REPLY macro instruction. Records are passed sequentially from the beginning of the queue entry.

Each record in the returned buffer is preceded by a eight-bytes prefix. Associated with each record, VSE/POWER provides an internal number in the following referred to as record number; this number must be specified in the restart control record when requesting to restart at that particular record.

If the 'CTLREC' option is specified in the SPL the SAS user task returns all records including immediate control commands to the user. Control commands are passed as one byte records, whereby the content of the byte is either blank or hex zero. The command code associated with the record is passed in the record prefix.

An output queue entry, in particular a LST queue entry may consist of multiple data sets; this is especially true when the job executed on a MVS system. The start of a new output data set or the change of one of the characteristics of a data set is recorded by VSE/POWER via a data set header record which precedes each data set. Since VSE/POWER attempts to keep all output data sets, produced by one job, together, multiple data set header records can appear in an output queue entry. Each data set header record encountered by the SAS user task is converted to a spool parameter list and passed as such to the cross-partition user.

If the cross-partition user indicated to 'quit' processing of the queue entry, the SAS user task, sets the appropriate cancel code in the XT-work area and calls the logical writer by means of the IPW$GLR macro instruction. This causes the logical writer to re-queue the queue entry with its original disposition, to write an spool account record and to free all acquired work spaces. The SAS user task then closes the logical interface by means of the IPW$CLI macro instruction and waits for further requests from the cross-partition user.

```
 USER                                 VSE/POWER
   |                                      |
OPEN:                               wait (RECB)
 request to access queue entry:           |
  SENDR (SPL) ────────────────────>       |
   |                                RECEIVE SPL
   |                                  • check SPL
   |                                  • open logical interface
   |                                  • locate queue entry
   |                                  • get JHR + DSHR and fill
   |                                    up SPL
 wait (SECB)                              |
   |  <──────────(extended SPL)───── REPLY
   |                                      |
   |   <.....................................
   |                                      |                 .
GETRQ:                               wait (RECB)            .
 request next buffer:                     |                 .
  SENDR  ──(user data)──────────────>     |                 .
   |                                      |                 .
 wait (SECB)                          • check buffer size   .
   |                                  • acquire buffer space.
   |                                  • fill up buffer      .
   |   <────(data buffer)─────────── REPLY                  .
   |                                      |                 .
   |       Loop till end of data (EOD in FDBK code)         .
   .........................................................
   |                                      |
 any finishing request:                   |
  SENDR  ──────────────────────────>      |
   |                                RECEIVE request
 wait (SECB)                              |
   |      <──────────────────(ok)───── REPLY
   |                                  • return queue entry
   |                                  • perform accounting
   |                                  • close log. interface
   |                                      |
   |                                  wait (RECB)
   |                                   • wait for next SPL
   |                                      |
 .....     any other activities         .....
```

Figure 123. GET Data/Control Flow

If the cross-partition user indicated to 'purge' the queue entry being accessed, the SAS user task, sets the appropriate cancel code in the XT-work area and calls the logical writer by means of the IPW$GLR macro instruction. This causes the logical writer to issue the IPW$DQS macro instruction with the purge indicator set, to write an spool account record and to free all acquired work spaces. The SAS user task then waits for further requests from the cross-partition user.

Likewise if 'close' of the queue entry is requested by the cross partition user, the SAS user task calls the logical writer to finish up that queue entry.

If the cross-partition user indicated that output processing failed, the SAS user task sets the appropriate cancel code in the XT-work area and calls the logical writer. The logical writer re-queues the queue entry with disposition "Y", writes an account record and frees all resources. This disposition prohibits the file from being processed by other tasks.

If the communication path is broken either intentionally or accidentally, the SAS user task returns the queue entry currently being processed to its class chain with its original disposition, closes the logical interface, performs accounting and detaches itself.

Restarting: While in the middle of receiving data from VSE/POWER, the cross-partition user can reposition at any record. If requested the restart function of the logical writer is utilized to locate the appropriate record. The restart can be based on records, lines or pages.

Checkpointing: A cross-partition user can request that checkpoint information be recorded for the queue entry currently in process. If such a checkpoint request is received by the SAS user task, the request is forwarded to the logical writer. The logical writer saves the user specified record number and copy number into the queue record. The queue record is then written back to disk.

Checkpointing with Extended Information: If the checkpoint information contains extended checkpoint information, the extended information is passed to the logical writer, as well. The logical writer writes the extended information to the queue control area.

During the 'open-process' for a queue entry the extended checkpoint information is retrieved from the queue control area using the macro IPW$IQS in order to pass the length of the extended information within the extended SPL to the cross-partition user. The retrieved extended information is held in storage till the next valid request comes from the cross-partition user, as it is assumed that usually the cross-partition user asks for the extended information after having received the extended SPL.

While in the middle of receiving data from VSE/POWER, the cross-partition user can retrieve the extended information. The macro IPW$IQS is used to read the extended information from the queue control area.

Obtaining OPTBs: A cross-partition user can request to get the OPTB structure associated by the queue entry currently in access. The request, passed to VSE/POWER via a control record, can be given at any time while accessing (GET function) a particular queue entry or while spooling output data (PUT function). If OPTBs are present in the data set header record, the SPL contains the total length of all OPTBs. If a 'get OPTB' control record is received by the SAS user task, the IPW$OPI FUNC=OPGET macro instruction is issued in order to get the wanted OPTBs. If an OPTB id is specified in the control record only this OPTB is moved into the cross-partitions user reply area. If, however, no OPTB id is specified, VSE/POWER moves all OPTBs into the requestor's area.

Modifying an OPTB: If a modify OPTB control record is received by the GET or PUT function routine, the "OPMOD" routine in IPW$$OP, is called by means of the IPW$OPI FUNC=OPMOD macro instruction to verify that the OPTB is correct. If so, the DBLK containing the data set header record is read in and the output processing section of the DSHR is updated with the new OPTB.

GET BROWSE Function: The SAS user requests obtaining data from the VSE/POWER queues by setting up and sending a Spool Parameter List (SPL) as described in “GET Function.” The following differences between GET and GET BROWSE exist:

1. GET BROWSE may access all queue entries regardless of their disposition, even queue entries 'active' or 'in creation' may be browsed.
2. The set of functions allowed for GET BROWSE is limited to
   - GET more data
   - RESTART
   - Obtain OPTBs
   - QUIT

Direct GET BROWSE for In Creation: Accessing a queue entry in creation is limited to job output. The spool Access Support application must set up a request as described in *VSE/POWER 6.5 Application Programming , SC33-6736-01* . For further information see "Get In Creation Queue Entry" on page 102.

Restarting to active record: Compared with GET an additional RESTART option (PXRSOPAR = Restart to active record) is available for GET BROWSE.

This request will reposition the cross-partition user at the last record passed by IPW$$GD to an UPDATE task, which processes currently the same queue entry. The request uses the fields QRCCNR, QRCCLC and QRCCPG of the queue record copy belonging to the UPDATE task to set up a restart request depending of the queue entry type. For output a restart to lines/cards is build using the value of QRCCLC. For jobs a restart to record is set up using the value of QRCCNR. The modified restart request is then passed to the restart function of the logical writer which is utilized to locate the appropriate record. In return either a return/feedback code will inform the requestor that no appropriate UPDATE task is active (on the same system) or a data buffer will be returned starting with the active record, which is followed by the subsequent records.

Note: The 4-byte data record number (QRCCLC for LST|PUN and QRCCNR for RDR type entries) is returned in 2 2-byte fields in the XPCCB (PXPLC12 & PXPLC34) for programs doing their own data record counting like ICCF.

PUT Function to RDR/XMT Queue: To start spooling a job into the VSE/POWER RDR or XMT queue an SPL must be sent to VSE/POWER specifying that job data follows.

Only the queue id 'R' has to be specified in the SPL. If the SAS user task receives an SPL, it calls the SPL parameter checker in order to check if the requesting userid is properly specified. If not, the appropriate return/feedback code is passed back and subsequently sent to the cross-partition user by means of the XPCC FUNC=REPLY macro instruction.

Otherwise the interface to the logical reader is opened via the IPW$OLI macro instruction. Each received data record in turn is then passed to the logical reader.

The SAS user task accepts records of any size, that means trailing blanks can be truncated by the cross-partition user before passing them to VSE/POWER. However, the SAS user task assumes that the logical record size is 80 bytes. If the user wishes to pass records larger than 80 bytes, the maximum record length must be specified in the SPL prior to passing the first record. The maximum record length supported by VSE/POWER is 128 bytes and the minimum record length is 80 bytes.

If the record is larger than the maximum length, specified in the SPL, the record is truncated. If a received record is smaller than the size, specified as maximum record length in the SPL, it is expanded into that length.

If the EOD indication is passed by the cross-partition user, the SAS user task invokes the logical reader to add the queue entry to the class chain according to its priority and to write a spool account record. When the job is not already at job boundary, which means the last record passed was not a job delimiter statement (neither * $$ EOJ nor /&), the SAS user task automatically adds the missing job delimiter depending on the input mode. When the queue entry is successfully queued in the queue, the SAS user task replies with an SPL, containing descriptive information about the job, such as defaults and job number assigned by VSE/POWER. When the final SPL is received by the user, it is the indication that VSE/POWER has taken full responsibility for the just submitted job stream.

If the spooled job is destined for a node which is not the LOCAL node, then it is queued in the XMT queue and not in the RDR queue. Figure 124 on page 359 illustrates the various steps done by VSE/POWER.

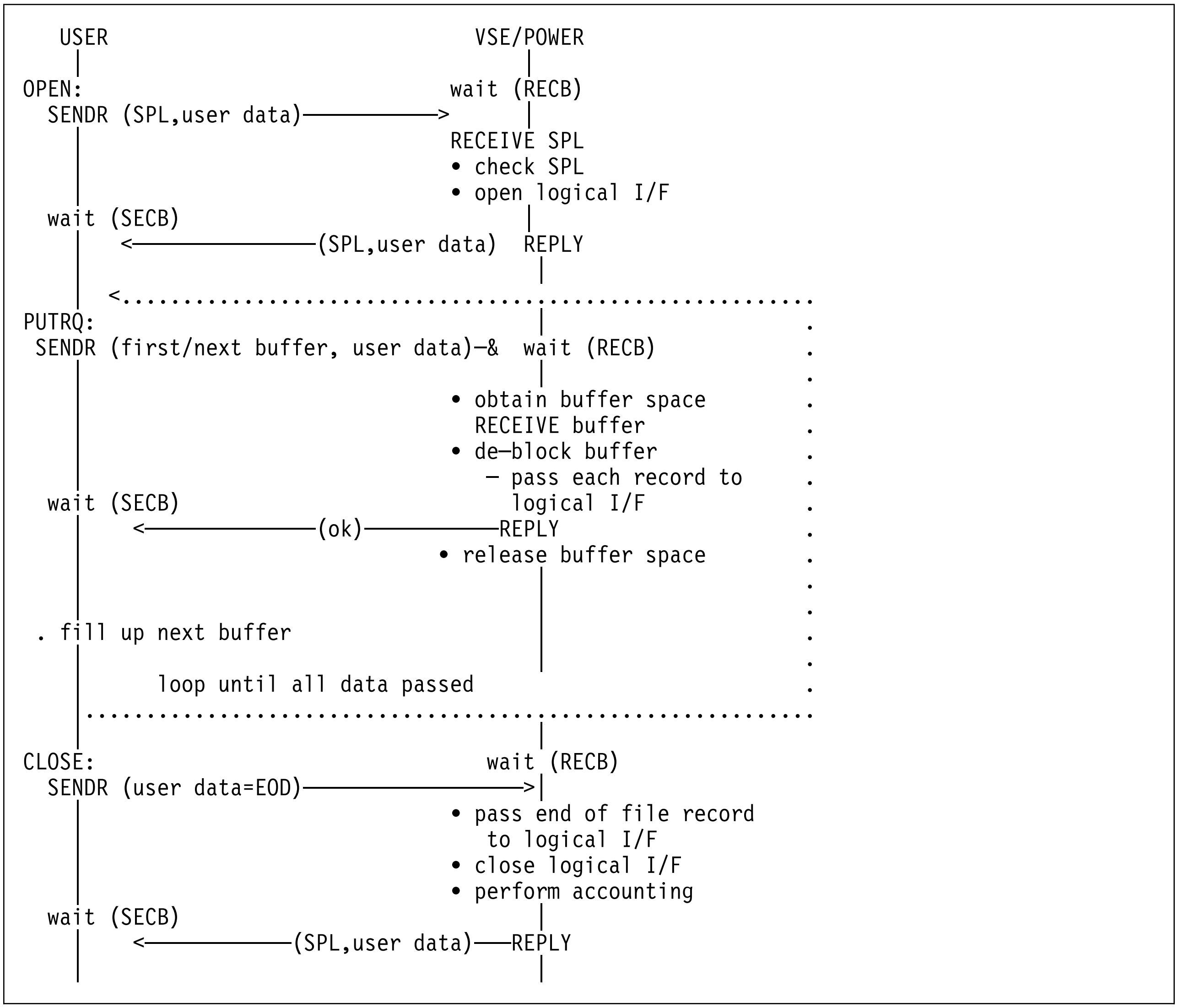
Note: When multiple VSE/POWER jobs are passed within one PUT function, the final SPL, sent back by VSE/POWER, reflects only the characteristics of the last job.

Note: VSE/POWER offers the user an option in the SPL to specify whether the job event message of the submitted job is to be queued by VSE/POWER rather than issued on the system console. For that purpose function byte SPLGFB1 has to be set up with SPLGF1QM (for generation of f.f. job completion messages), or SPLGF1QQ (for generation of job event messages, which are both f.f job completion and job generation messages). The submitted job is then flagged by VSE/POWER in its queue record and in the VSE/POWER section of the job header record. The job processing VSE/POWER system inspects these flags after the job finished its processing, and builds a Nodal Message Record from the resulting job completion message or as soon as the job has been created in IPW$$XWE.

With the information contained in the record prefix of the nodal message the final message retrieving VSE/POWER system then decides whether the message has to be queued or not.

The messages can then later be retrieved by the GCM Function of the Spool-access support.

There exist some options that can be specified within the PUT-OPEN SPL, for example, additional private information, queue type (COMMON queue or single userid queue). For these options consult the manual VSE/POWER Application Progamming.



```
   USER                              VSE/POWER
    |                                     |
OPEN:                              wait (RECB)
  SENDR (SPL,user data)———————————>      |
    |                              RECEIVE SPL
    |                              • check SPL
    |                              • open logical I/F
  wait (SECB)                             |
    |     <————————————————(SPL,user data)  REPLY
    |                                      |
    |    <..........................................................
PUTRQ:                                     |                      .
 SENDR (first/next buffer, user data)—&  wait (RECB)              .
    |                                      |                      .
    |                              • obtain buffer space          .
    |                                RECEIVE buffer               .
    |                              • de—block buffer              .
    |                                 — pass each record to       .
  wait (SECB)                           logical I/F               .
    |      <———————————————(ok)———————————REPLY                   .
    |                             • release buffer space          .
    |                                      |                      .
    |                                      |                      .
 . fill up next buffer                     |                      .
    |                                      |                      .
    |       loop until all data passed                            .
    |.............................................................
    |                                      |
CLOSE:                                wait (RECB)
  SENDR (user data=EOD)———————————————————>|
    |                              • pass end of file record
    |                                 to logical I/F
    |                              • close logical I/F
    |                              • perform accounting
  wait (SECB)                              |
    |      <—————————————(SPL,user data)———REPLY
    |                                      |
```

Figure 124. PUT Function Control/Data Flow

PUT Function to LST/PUN or XMT Queue: To start spooling output (list or punch) into the VSE/POWER LST/PUN queue, an SPL, describing the attributes of the output, must be sent to VSE/POWER by issuing the XPCC FUNC=SENDR macro instruction before any data transfer. The output format may be ASA, machine control code, SCS print, 3270 data stream or escape. Special formatting, such as graphics, is catered by the escape mapping, which allows the user his own output formatting.

If the SAS user task receives an SPL, requesting spool output service, it calls the SPL parameter checker to verify that all specified fields in the SPL are valid. If an invalid or missing specification is detected, the SAS user task rejects the request and passes appropriate return and feedback codes back to the cross-partition user.

If the SPL was correct, the interface to the logical output spooler is opened by means of the IPW$OLI macro instruction. The logical routine is then called via the IPW$PLR macro instruction to acquire spool space and work areas used for the job header and data set header record. Next, the job header record and data set header record are constructed from the information extracted from the SPL and written to disk. The data set header record consists of the general and VSE/POWER section. If 3200/3800 parameters are defined in the SPL, a 3800 section is appended to the data set header record.

If an OPTB area, consisting of one or more output parameter text blocks, is appended to the SPL, the VSE/POWER SAS user task calls the 'OPPUT' routine by means of the IPW$OPI FUNC=OPPUT macro instruction to check if the OPTBs are properly built. Furthermore, all OPTBs which are defined in the OPDE chain are examined for correct specification. An OPTB representing a keyword which is not defined within VSE/POWER is taken as is. If an OPTB area is present and all OPTBs are valid, an output processing section is built and included in the data set header record.

If the OPTB area, appended to the SPL consists of keyword OPTBs the VSE/POWER SAS user task calls the 'OPANAL' routine by means of the IPW$$OPI FUNC=OPANAL macro instruction. For each keyword OPTB an output parameter text block is built according the specifications of the corresponding DEFINE statement. The keyword value is check and if correct the output parameter text block is included into the data set header record. Invalid keywords are not included into the data set header record instead the request is terminated with return/feedback code.

The OPTB is structured as a sequence of text units. The number of and sequencing of text units is arbitrary. Text units are identified by a 2-byte id that is registered and unique within job networking protocols. Each text unit is defined to include a specific type and maximum number of data elements that represent keyword parameter values. The number of these data elements included in the text unit is specified in a 2-byte count field that follows the id and precedes the data elements. The figure below illustrates this structure.

```
         ┌─────┬────┬────┬──────────────┬────┬──────────────┬──//──────┐
         │ ID  │ CC │ LL │ data element │ LL │ data element │          │
         └─────┴────┴────┴──────────────┴────┴──────────────┴──//──────┘
Bytes:     2     2    2         n          2         n

where:
    ID  —  registered keyword identifier
    CC  —  number of data elements
    LL  —  length of data element
```

Figure 125. Output Parameter Text Block Structure

VSE/POWER assigns a new job number to the output queue entry. This number is returned in the SPL. The 'verification' SPL is then returned to the cross-partition user by means of the XPCC FUNC=REPLY macro instruction. The SAS user task waits then for output data to arrive from the cross-partition user.

As VSE/POWER receives each buffer, checks are made to ensure that the buffer size does not exceed the VSE/POWER limited maximum length of 64K and that each individual record does not exceed the maximum length of 32K - 1. If the maximum buffer length is exceeded, appropriate return and feedback codes are set and the communication path is unconditionally terminated via the XPCC FUNC=DISCPRG macro. Records may be sent as one record per buffer or as multiple records per buffer.

The SAS user task does not check the validity of any carriage control character which might be associated with a data record. The specification of a carriage control character for BMS, 3270, SCS or escape mode is ignored. The X'FF', X'FE' and X'FD' carriage control character are reserved for VSE/POWER usage only and can therefore not be used by a cross partition user.

Each record in turn is passed to the logical output spooler. As VSE/POWER spools each record, page and line counts are maintained. Page breaks are determined from the carriage control character associated with each record passed. The meaning of the carriage control character (either MCC or ASA) is defined at 'open' time in the SPL. The line count is updated according to the carriage control character. For BMS or 3270 mapping, each record is assumed to be a page. For escape and SCS data stream, the page count is meaningless and therefore set to zero. The line count is also undetermined, but set to the current spooled record number.

If a CPDS data record should be spooled, it must be indicated as such in the identification byte of the record prefix.

Record Format	Line Count	Page Count
MCC	Updated according to carriage cntl character.	Updated according to carriage cntl character.
ASA	Incremented for each record.	Updated according to carriage cntl character.
BMS mapping	Incremented for each record.	Incremented for each record.
3270 mapping	Incremented for each record.	Incremented for each record.
Ecape mode ESC	Incremented for each record.	Set to zero.
SCS mode	Incremented for each record.	Set to zero.
CPDS mode	Incremented for each record.	-
CPDS records intermixed with ASA/MCC records	Incremented for each see MCC, ASA record.	Updated according ASA/MCC carriage cntl character

Any invalid request or conflicting requests, as indicated in the action bytes of the passed user data of the XPCCB, cause appropriate return/feedback code to be set, and the request to be ignored.

If the EOD indication is passed by the cross-partition user, the SAS user task invokes the logical output spooler to add the queue set to the class chain according to its priority and forms number and to write a spool account record.

If together with the EOD indicator an SPL is sent by the cross partition user, certain fields in the SPl are used to update some characteristics of the output which has just been spooled. If the output is destined for a remote node, it is queued into the XMT queue, otherwise in one of the local queues (LST or PUN). The logical interface is then closed via the IPW$CLI macro instruction. When the queue entry is successfully queued, the SAS user task replies with an SPL, containing final information such as job and job suffix number. If messages are queued by VSE/POWER, the SAS user task sets also the 'message queued' flag in the information byte of the user data of the XPCCB. The cross-partition user can then retrieve the messages. If, however, a 'quit' request is received, the current queue entry being built is purged, its already used spool space is returned to the free data file space, the logical interface is closed and the SAS user task waits then for new requests from the cross-partition user.

Output Segmentation: At any time while spooling output, a segment request may be submitted to VSE/POWER either with a null buffer or together with data.

When such a request is received the SAS user task processes the data received with the segmentation request, if any, then adds the queue entry to the class chain according to the normal priority rules. After the queue entry is added new spool space is reserved in order to accept further spool requests for the same output. All queue entries for this output will have the same job number but a different job suffix.

RESTART during PUT request for output: While spooling output, the user may request to re-position at a previous spooled record and to continue spooling from that point. This is done by sending a restart control record to VSE/POWER. The restart control record contains the logical record number from where to begin.

Checkpoint processing for output spooling: For the situations of:

- User Abend
- VSE/POWER Abend
- System Abend

the "checkpoint" PUT function is offered.

While spooling Output, the cross-partition user may request the "checkpoint" function to occur. The records held in storage are written to disk and a "checkpoint" ID is passed back. The queue record is updated with the number of the last record written to the data file. This number becomes the "checkpoint" ID.

When a checkpoint request is sent together with a data buffer, the data buffer is processed before processing of the checkpoint request.

When a checkpoint is successfully recorded, the SAS user task sends back a checkpoint response record.

If the user abnormally terminates the connection during the PUT function or the SAS user task is PSTOPed, then the Output data set is closed with a job trailer record being added, if possible, to the end of the data and given the disposition "X" and written to disk and placed in the class chain as for a complete file. This disposition prohibits the file from being processed by other tasks. When the user has resumed operation he may then access the data with the "restart" PUT function, specifying the "checkpoint" ID which indicates to VSE/POWER where the spooling is to resume.

If VSE/POWER or VSE/AF abends, then during recovery at startup time, VSE/POWER will search the queue file for incomplete queue entries. If found and the queue entry was checkpointed previously, then VSE/POWER will reposition the queue entry at the last checkpoint record and then attempt to write a job trailer record. If no spool space is available to write the job trailer record, just an end-of-data flag is set. The data set disposition is set to "X" and written to disk to be placed in the class chain later. Then the user may perform the "restart" step as above. Any DBLK group in the queue entry following the new end-of-data record is unchained and placed in the free DBLK group chain.

GCM Function

Overview VSE/POWER's GCM Function of the Spool-access support is used to retrieve job event messages of jobs by a user written application program, provided that the jobs have been submitted via the Spool-access support function PUT and function byte SPLGFB1 has been set up with SPLGF1QM or SPLF1QQ. The GCM function offers the user various requests and subrequests for

- retrieving job event messages and
- manipulating the contents of the fixed format message queues

For the storage location of the message queues and the data relationship see Figure 47 on page 134.

Each request - in the following referred to as GCM-OPEN request -, uses an SPL to pass information to VSE/POWER. All following subrequests use also this information, which is for example:

- The XPCC application ID and the PWRSPL user-id to address the message
- The message selection criteria specified by job name and job number in the PWRSPL.
- The request type specified in SPL-function byte SPLGFB1.
- Several optional specifications in the SPL (e.g. SPLXWAIT).

Retrieving Job Event Messages: In order to retrieve job event messages two requests types are offered:

1. GCM-OPEN-KEEP with the GCM-MORE and the GCM-REMOVE subrequests
2. GCM-OPEN-DELETE with the GCM-MORE subrequest

These GCM-OPEN requests may also be specified along with a wait interval in field SPLXWAIT. The request will then wait at most so many seconds as specied or will return with the message, if available. A value from 0 to 27962 seconds can be specified. If the specified value is greater, the request will wait for ever. If the message is available, the application is then posted.

GCM-OPEN-KEEP: This requests a copy of the messages queued at the addressed message queue. All retrieved messages may later be retrieved again.

GCM-OPEN-DELETE: This request type retrieves messages queued at the relevant message queue. All retrieved messages are deleted at the moment of retrieval.

Manipulating the Contents of the Message Queues: In order to delete messages from a queue the GCM-OPEN-REMOVE or the GCM-OPEN-PURGE requests can be used. These requests offer a wider range of possibilities to delete job event messages than the GCM-REMOVE subrequst provides. Because the request is issued together with an SPL, the user may

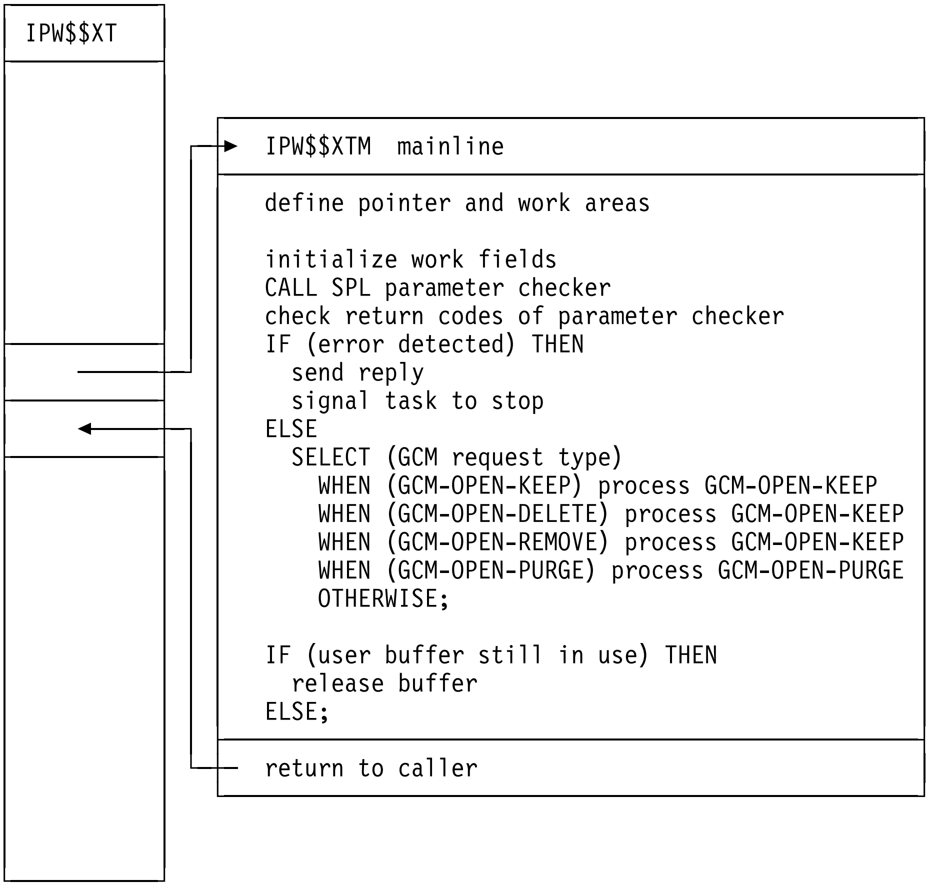
- alter the user-ID and the XPCC application-ID
- change the selection criteria

In contrast to the GCM-REMOVE subrequest this request will delete **all** job event messages which have already been retrieved by **any** preceding GCM-OPEN-KEEP request(s) and which match the selection criteria.

Function Logic: The overall program logic of the GCM function is shown in Figure 126 on page 364. Function module IPW$$XTM is called by the cross partition user task, when a new SPL is received. After pointers and work areas have been established, the parameter checker IPW$$PC is called. If any error has been detected by IPW$$PC, an XPCC SENDR request is issued to inform the user about the incorrect specified SPL. The task-to-stop indicator is set up and control is returned to IPW$$XT. Otherwise the specified GCM request is processed. The GCM request is terminated if

- a new SPL occurs
- the user issued XPCC DISCPRG
- an error occurred

The reply buffer is then released and control is returned to IPW$$XT (Figure 126).



```
IPW$$XTM  mainline

define pointer and work areas

initialize work fields
CALL SPL parameter checker
check return codes of parameter checker
IF (error detected) THEN
  send reply
  signal task to stop
ELSE
  SELECT (GCM request type)
    WHEN (GCM-OPEN-KEEP) process GCM-OPEN-KEEP
    WHEN (GCM-OPEN-DELETE) process GCM-OPEN-KEEP
    WHEN (GCM-OPEN-REMOVE) process GCM-OPEN-KEEP
    WHEN (GCM-OPEN-PURGE) process GCM-OPEN-PURGE
    OTHERWISE;

IF (user buffer still in use) THEN
  release buffer
ELSE;

return to caller
```

Figure 126. GCM Function Program Logic

Flagging Mechanism

GCM-OPEN-KEEP and GCM-MORE/GCM-REMOVE: At the beginning of a GCM-OPEN request, all flag 2's of the relevant message queue are removed. Then the specified request type is processed, which is in this case a GCM-OPEN_KEEP request. Any retrieved message is copied from the message queue to the user's reply buffer and marked at the point of retrieval with two flags, flag 1 and flag 2 (Figure 127 on page 365 a). Flags of type 1 are never removed, flags of type 2, however are removed with the beginning of the next GCM-OPEN request. A flag of type 1 is relevant for a later GCM-OPEN-REMOVE request, a flag of type 2 is relevant for GCM-MORE and GCM-REMOVE subrequests.

Any new GCM-MORE request scans the message queue from the beginning of the queue whereby all relevant messages are skipped which are flagged with flag 2 (Figure 127 on page 365 b).

Any relevant unflagged message is flagged with flag 1 and flag 2 and copied to the user' s reply buffer until all relevant messages are retrieved or the user's reply buffer is full (Figure 127 on page 365 c).

Finally, when all relevant messages are retrieved, a final GCM-REMOVE subrequest can be issued by the application which then removes all messages flagged with flag2 (Figure 127 on page 365 d).

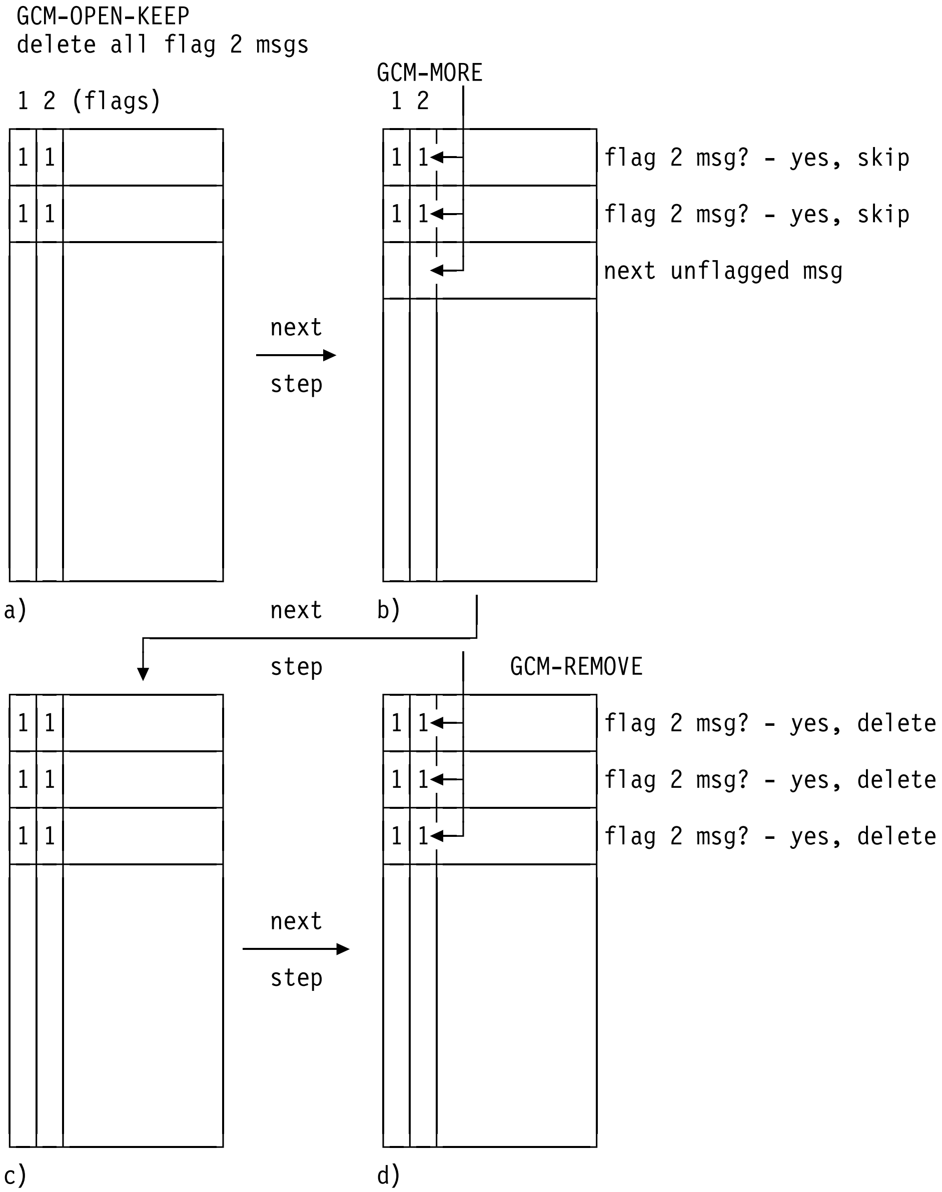


Figure 127. Processing GCM-OPEN-KEEP Followed by GCM-MORE and GCM-REMOVE

Flagging Mechanism with GCM-OPEN-KEEP and GCM-OPEN-REMOVE: Flag 1's are never removed. This means that more and more messages of a specific message queue are flagged with flag 1 during the processing of several GCM-OPEN-KEEP requests. Messages flagged by the current GCM-OPEN-KEEP request are marked with 'A', messages retrieved by a preceding GCM-OPEN-KEEP request are marked with 'B' in Figure 128 on page 366. A GCM-OPEN-REMOVE request scans the message queue for relevant messages flagged with flag 1 and removes all of them (Figure 128 on page 366 d)) in contrast to the GCM-REMOVE request, which only deletes messages of request A.

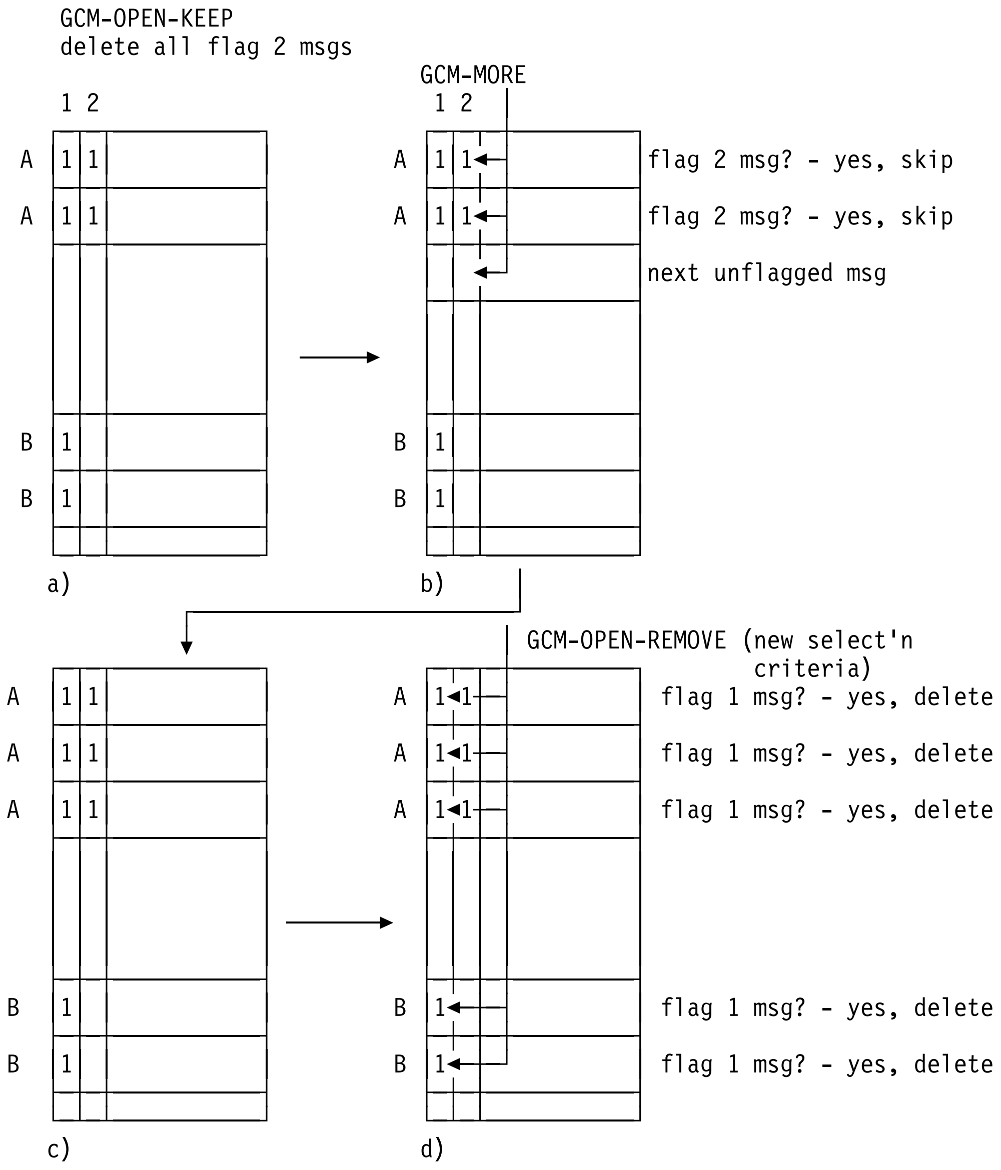

GCM-OPEN-KEEP
delete all flag 2 msgs
GCM-MORE
1 2
1 2
A
A
B
B
flag 2 msg? - yes, skip
flag 2 msg? - yes, skip
next unflagged msg
a)
b)
GCM-OPEN-REMOVE (new select'n criteria)
flag 1 msg? - yes, delete
flag 1 msg? - yes, delete
flag 1 msg? - yes, delete
flag 1 msg? - yes, delete
flag 1 msg? - yes, delete
c)
d)

Figure 128. Processing GCM-OPEN-KEEP Followed by GCM-OPEN-REMOVE

GCM-OPEN-PURGE Request: The request deletes all messages from one or more fixed format message queues.

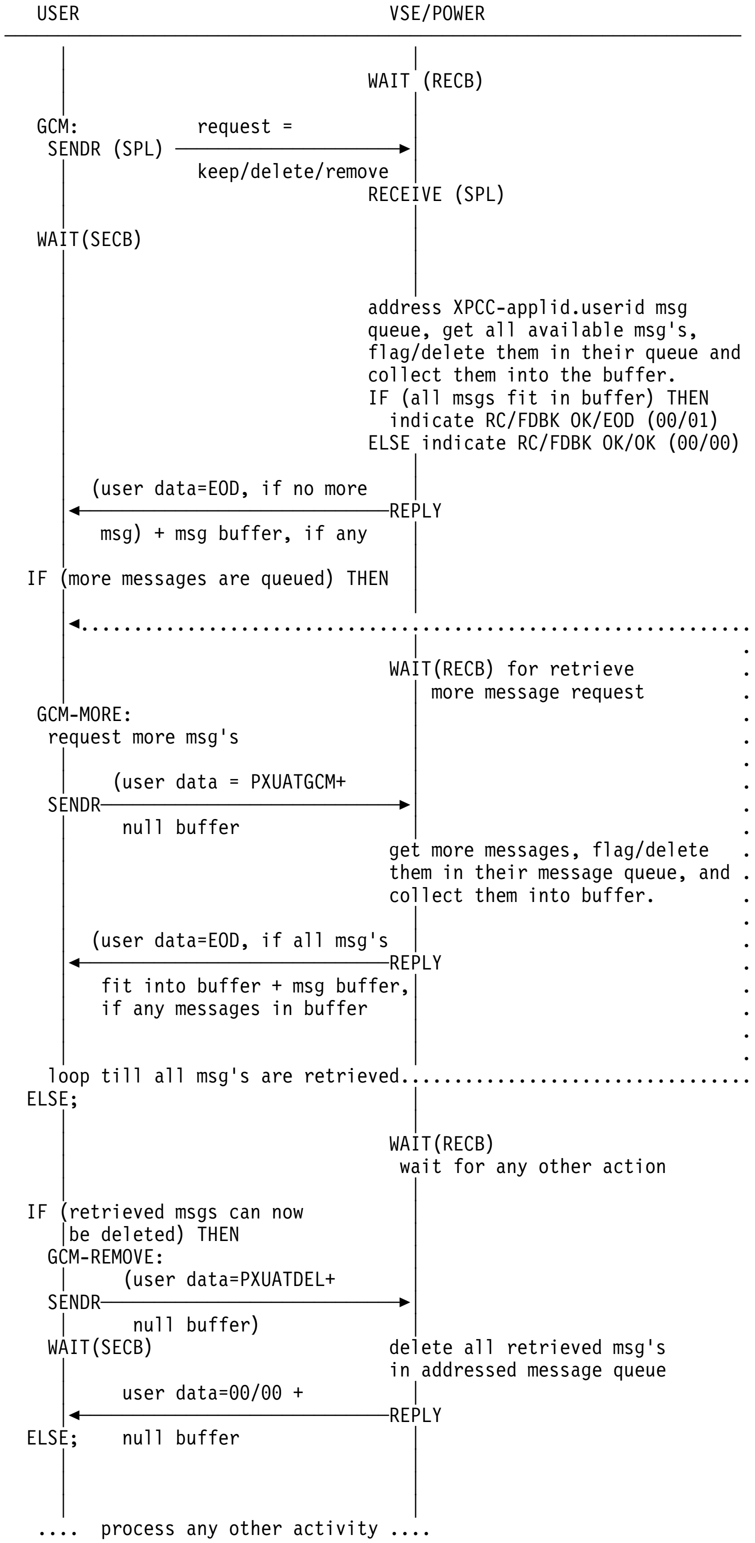
Depending on the user's specifications in fields SPLGUS and IJBXTOAP, the request will operate on one ore more queues. If field SPLGUS contains 8 blank characters as generic userid, all queues with the same contents in field ACIEAPPL and which match also the specification in field IJBXTOAP, will be processed.

If field SPLGUS contains any other alphameric value, only a specific queue will be processed.

A 'blank' SPLGUS userid may only be specified for a GCM-OPEN-PURGE request.

Any wait specification in field SPLXWAIT is ignored for this request.

Interaction Between User Program and VSE/POWER



```
   USER                             VSE/POWER
 ______________________________________________________________________
     |                                |
     |                             WAIT (RECB)
     |                                |
   GCM:           request =           |
    SENDR (SPL) ----------------------->
     |            keep/delete/remove  |
     |                             RECEIVE (SPL)
     |                                |
   WAIT(SECB)                         |
     |                                |
     |                             address XPCC-applid.userid msg
     |                             queue, get all available msg's,
     |                             flag/delete them in their queue and
     |                             collect them into the buffer.
     |                             IF (all msgs fit in buffer) THEN
     |                               indicate RC/FDBK OK/EOD (00/01)
     |                             ELSE indicate RC/FDBK OK/OK (00/00)
     |                                |
     |  (user data=EOD, if no more    |
     |<-----------------------------REPLY
     |   msg) + msg buffer, if any    |
     |                                |
  IF (more messages are queued) THEN  |
     |                                |
     |<.......................................................
     |                                |                      .
     |                              WAIT(RECB) for retrieve  .
     |                                |  more message request .
   GCM-MORE:                          |                      .
    request more msg's                |                      .
     |                                |                      .
     |    (user data = PXUATGCM+      |                      .
    SENDR------------------------------>                     .
     |     null buffer                |                      .
     |                              get more messages, flag/delete   .
     |                              them in their message queue, and .
     |                              collect them into buffer.        .
     |                                |                      .
     |  (user data=EOD, if all msg's  |                      .
     |<-----------------------------REPLY                    .
     |   fit into buffer + msg buffer,|                      .
     |   if any messages in buffer    |                      .
     |                                |                      .
    loop till all msg's are retrieved.........................
  ELSE;                               |
     |                                |
     |                              WAIT(RECB)
     |                               wait for any other action
     |                                |
  IF (retrieved msgs can now          |
     |be deleted) THEN                |
    GCM-REMOVE:                       |
     |     (user data=PXUATDEL+       |
    SENDR------------------------------>
     |      null buffer)              |
    WAIT(SECB)                      delete all retrieved msg's
     |                              in addressed message queue
     |     user data=00/00 +          |
     |<-----------------------------REPLY
  ELSE;    null buffer                |
     |                                |
     |                                |
   ....  process any other activity ....
```

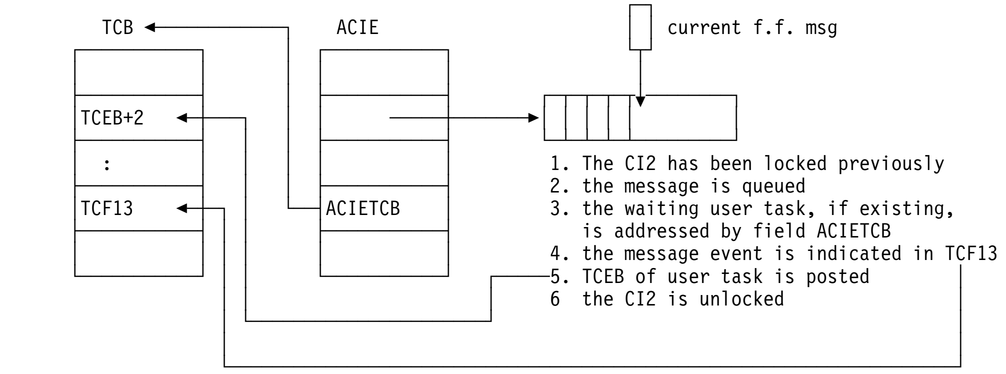
Figure 129. GCM Function Control/Data Flow

Specification of Wait Intervals - Posting:

Waiting GCM user task | Message generating task

- execution reader task
- execution writer task
- timer task
- network receiver task

currently queuing the fixed format message to the message queue in IPW$$NS



1. The CI2 has been locked previously
2. the message is queued
3. the waiting user task, if existing, is addressed by field ACIETCB
4. the message event is indicated in TCF13
5. TCEB of user task is posted
6 the CI2 is unlocked

o When the waiting user task is terminated, the address of its TCB is cleared in the ACIE in field ACIETCB. Access serialization is controlled by locking and unlocking the CI2.

o POWER main task:
As soon as the posted TCEB is detected by the VSE/POWER task dispatcher, the user task is dispatched and the event is analysed (in IPW$$XTS, IPW$$XTM):

```
  IF a message is queued:
   - the wait interval is cleared
   - the message is passed to the user
   -a wait is set up for the next request.
  IF other event is detected:
   - analyse event and handle
     accordingly (PEND, PSTOP,..)
```

Figure 130. GCM Function Control/Data Flow (cont.)

Issue Job Completion Message Due to PRELEASE.: It may occur that jobs are in the reader queue which have been submitted without the option 'issue completion message'. For example jobs read in via the local reader or received from non-VSE/POWER nodes do not have this option, because the option 'issue completion message' can be used only, if jobs are submitted via the spool-access-support. In order to get a completion message even for this kind of jobs, an option can be used when a PRELEASE command is issued via the spool-access-support. This option is the same option which can be used when a job is submitted to the reader queue using the PUT service of the spool-access support.

Main Rules For Issuing Completion Messages To The Releaser

1. A completion message for the releaser has the same layout as a completion message for the submitter. No indication is set whether the completion message has been issued for a releaser or a submitter.

   But a completion message for a releaser contains always the original jobnumber (JCMFONUM) which is the number of the job at the time the PRELEASE command has been issued. This original jobnumber is useful if a job is sent to another node for execution, because usually a job gets a new jobnumber on another node. If the job is executed at the same node where the PRELEASE command has been issued, the original jobnumber(JCMFONUM) is the same as the jobnumber (JCMFNUM).

2. A completion message for the releaser is issued no matter if a completion message for the submitter has to be issued or not.
3. A completion message for the releaser is issued in addition to a completion message for the submitter.
4. If the releaser is the same as the submitter, only one completion message is issued.

   In this case the completion message issued is the message for the submitter, which means it may contain an original jobnumber (JCMFONUM) or not, depending on the option used at the time the job was submitted.

   The releaser is the same as the submitter, if both use the same application-id and user-id, and if both are running on the same system, which means node-id and system-id (of a shared complex) are the same.
5. A completion message for the releaser is issued only if the PRELEASE command was processed 'successfully'.
   If the message 1R88I NOTHING TO RELEASE has been issued, the PRELEASE command was processed unsuccessfully, which means a completion message for the releaser is not issued.
   The message 1R88I NOTHING TO RELEASE is issued for example, if the job to be released has already a disposition of D or K.
6. The attribute 'issue a completion message for the releaser' can not be reset, neither by a PALTER nor a PHOLD nor any other command.
7. If more than one PRELEASE command with the indication 'issue a completion message to the releaser' is used, only one completion message to the releaser is issued using the application-id and user-id of the last successful PRELEASE command. Note that this may happen only in special situations, for example if a PHOLD or PALTER command has been issued to set the job in non-dispatchable state before the job has been executed.
8. The attribute 'issue a completion message for the releaser' is a temporary one.
   a. It is no longer available, if the job has been executed.
   b. It is no longer available, if the POFFLOAD command is used, to write the job to tape.
   c. It is not inherited to a child job which has been created by a parent job (for example when a job is created by using the DISP=I operand within the * $$ PUN statement when a job submits another job using the spool-access-support).
9. In a shared environment a completion message for the releaser is routed back to the system on which the PRELEASE command was issued.

   If the PRELEASE command was issued twice, although on different systems, only one completion message for the releaser is issued, namely to that system which issued the last 'successful' PRELEASE command.
10. In a network a completion message for the releaser is routed back to the node on which the PRELEASE command was issued.

    Rules for jobs within a network:

    a. The attribute 'issue a completion message for the releaser' is sent together with a job through a network.

       If a job is sent from node A to node B, the completion message for the releaser is sent back from node B to node A, if a PRELEASE command was issued on node A as long as the job was on node A.

       This completion message for the releaser is sent back from node B to node A only once. If the job is executed on node B once more, no second completion message for the releaser is sent back to node A.

b. The attribute 'issue a completion message for the releaser' is not sent together with the PRELEASE command through a network.

   A completion message for the releaser is not sent back from node B to node A, if the job has been sent from node A to node B and thereafter the PRELEASE command is sent from node A to node B and the PRELEASE is executed on node B.

c. Only one occurrence of the attribute 'issue a completion message for the releaser' is sent together with a job through a network.

   Consider the following example: a PRELEASE command is issued on node A and thereafter the job is sent to node B. Another PRELEASE command is issued on node B and thereafter the job is sent to node C. If the job is executed on node C, a completion message for the releaser is sent to node B which is the last node where the PRELEASE command was issued before the job was sent to another node.

d. Two 'completion messages for the releaser' may be issued:
   1) one 'completion messages for the releaser' for the local node
   2) one 'completion messages for the releaser' for the sending node

   This happens in the following case:
   A PRELEASE command is issued on node A and thereafter the job is sent to node B. Another PRELEASE command is issued on node B and thereafter the job is executed on node B. A completion message for the releaser is sent to node A and is also issued for the releaser on the local node B.

   Only one completion message for the releaser is issued, if the job is sent back from node B to node A, and on node A another PRELEASE command is successfully processed. The completion message for the releaser is issued due to the last PRELEASE command which means the original jobnumber(JCMFONUM) is the same as the jobnumber (JCMFNUM).

e. If a job is sent to another node for execution and is written to tape using the POFFLOAD command and read in later on again using the POFFLOAD command, the attribute 'issue a completion message for the releaser' has been lost, and no 'completion message for the releaser' is issued. This happens for any completion message: no message is issued for a releaser on the local node and no message is issued for a releaser on a remote node. This happens always, no matter on which node the job has been written to tape: either on the original node or the execution node or any intermediate node.

Rules concerning the original jobnumber:

a. If a job is sent to another node and back to the 'original' node where the PRELEASE command has been issued and is now executed on the 'original' node, the original jobnumber (JCMFONUM) is usually not the same as the current jobnumber (JCMFNUM), because sending jobs to other nodes usually assigns new jobnumbers.

b. It may occur that the original jobnumber (JCMFONUM) of a message for a submitter is not the same as the original jobnumber (JCMFONUM) of a message for a releaser, if the job was submitted on node A, sent to a node B and has been released on node B. The original jobnumber of a message for a submitter is the jobnumber on node A where the job has been submitted. The original jobnumber of a message for a releaser is the jobnumber on node B where the job has been released.

Special Situations For Completion Messages: The following describes a few 'special' situations.

1. The attribute 'issue a completion message for the releaser' is a temporary one and is lost as soon as the job has been executed completely.

   a. If a job runs several times due to DUEDAY operands, a completion message for the releaser is issued only once, if a PRELEASE command was successfully issued once.

2. The attribute 'issue a completion message for the releaser' stays until the job has been executed completely. No completion message for the releaser is issued and the attribute is not changed:

   a. If the job's execution has not yet started and
      1) If VSE/POWER is terminated normally (via the PEND command) or abnormally (for example due to a program check).
      2) If the PALTER command is used to change the disposition to H or L and later on back again to D or K.
      3) If the PALTER command is used to change the disposition to H or L and if later on the PRELEASE command is used without the indication 'issue message to releaser'.
   b. If the job's execution has started and
      1) If VSE/POWER is terminated abnormally.
      2) If recovery is done for the processing system using the PRESET comand.

3. Details for jobs within a network:

   The completion messages are sent back from the execution node to the original node. But the completion messages is issued only if

   a. the execution node is a VSE/POWER node
   b. the execution node has the 'correct level' of VSE/POWER (version 6.1 and later, which means VSE/ESA 2.1 and later).
   c. the path via the network is available, which means all the nodes concerned with passing the message must be 'signed on'.

Interaction A completion message for the releaser is issued, if:

1. a PRELEASE command (free fromat or not) is sent via the CTL-service
2. the field SPLGFB2 is set to SPLGF2MR
3. the field SPLGFB1 is set to SPLGF1QM

The destination of a completion message is defined by:

1. the XPCC-application-id of the CTL-service
2. the user-id of the CTL-service
3. in addition without having to be specified by the user
   a. the system-id in an shared environment
   b. the node-id in a network

A completion message for the releaser may be retrieved using the GCM-service the same way which is used to retrieve a completion message for the submitter.

SPLGF2MR can not be set using the FUNC operand of the PWRSPL macro.

SPLGF1QQ, which causes to issue job event messages, is not supported.

Messages and Codes No special messages or codes exists:

1. Validation of XPCC-application-id is done as for all other cases.

2. Validation of user-id is done as for all other cases.

3. Validation of function byte is done as for all other cases.
   SPLGFB2 is not validated (if set during a GET- or PUT-service).

4. layout of a completion message for the releaser is the same layout of a completion message for the submitter (with the exception of the original jobnumber, see above 'Main Rules For Issuing Completion Messages To The Releaser').

Documentation This support is for 'internal use' only, which means it is only to be used by REXX. It is described only in the VSE/POWER Diagnosis Reference Manual. It is not described in the VSE/POWER Application Programming.

Implementation CAF-code used is @D61IDHS and @KXA1255.

PWRSPL: The function code SPLGF2MR is defined to issue completion message to the user who issued PRELEASE command.

There is no support to validate SPLGF2MR, neither if it is set for any other (GET,PUT,GCM) service (where it is just ignored) nor if this bit is set and SPLGF1.. is missing or invalid (also no completion message to the releaser will be issued).

No validation is done neither by IPW$$PC nor by IPW$$XTC. Only IPW$$CR tests for SPLGF2MR, and expands its meaning into the queue record.

IPW$DQR and IPW$DQC.

1. Following fields are defined to identify the releaser:
    a. QRMRSI containing system_id (for shared)
    b. QRMRUS containing user_id
    c. QRMRAP containing application_id
2. QR-OP2 contains O2MR indicating 'issue completion message to releaser'
3. QR-OP2 contains O2MQ indicating 'issue completion message to releaser using info out of queue-record', because info out of job header record has been used 'once' and should not be used a second time.

    This indication is used in the following cases:

    a. Job J is 'PRELEASED' at node A and sent to node B and at node B the job J is 'PRELEASED' twice, but only for the first PRELEASE a message is sent back to A to be consistent with local function.
    b. Job J is 'PRELEASED' at node A and sent to node B and at node B the job J is written to tape. If the job J is spooled back to the reader queue from the tape and executed, no completion message should be issued in order to be consistent with 'local message issuing'. In order to avoid lots of code to clear the information in the job header record, the bit QRO2MQ is used.

IPW$DNR Following fields are defined to identify the releaser:

1. NJHPMRAP containing application_id
2. NJHPMRUS containing user_id
3. NJHPMRND containing node_id
4. NJHPMRSI containing system_id (for shared)
5. NJHPMRON containing original jobnumber

IPW$DTC A interface bit between IPW$XRE and IPW$$MS is defined to use correct application_id when building a completion message for the releaser.

IPW$$CR: The function codes SPLGF1QM and SPLGF2MR are processed near label PRELS20: If both used, update queue-record

1. with indication 'issue completion msg due to PRELEASE'
2. with user-id out of SPL
3. with appl-id out of XTWAREA
4. with system-id out of DMB

If just one of the 2 function codes SPLGF1QM and SPLGF2MR is used, no completion message is issued and no error return code is set.

IPW$XRE: The completion message for a releaser is issued just once. Once a completion message for a releaser is issued, the information thereabout must be updated and written to the queue file. Therefore the completion message for a releaser is issued before a queue record is written back to the queue file using the macros IPW$$DQ and IPW$$FQ near label XQ94 and not at the label where the completion message for a submitter is issued.

If a completion message for the releaser has to be issued, use existing macro IPW$NTY with existing parameter 'GCM' to issue completion message but pass appropriate information for user_id, node_id, and system_id:

1. If information for issuer can be found in queue-record, take the information out of the queue-record.
2. If information for issuer can be found in job header record take the information out of the job header record.

The application_id for the completion message is chosen by module IPW$$MS which builds the completion message. Use new parameter TC15MR to signal IPW$$MS to retrieve the application_id for the releaser.

If information for a releaser can be found in the job header record and the queue record, make sure the completion message is sent to both releasers, one to the other node and one to the local node. Set indication that information out of job header record has been used to issue completion message to avoid a second completion message in the case the job is executed a second time. This indication is set in the queue record, because the job header record can not re-written to the spool file.

Having issued a completion message due to the information in the queue record, reset information in queue record (unconditionally).

If the releaser and the submitter are the same (same user_id, same application_id, same node_id, same system_id), omit message for the releaser, but issue only the message for the submitter.

IPW$$MS: When a completion message is built near label MSNMR04 TC15MR and QRO2MR are used to retrieve the appropriate application_id for the completion message. Both indications are set in IPW$XRE.

IPW$$NT: The information about completion message in job header record is updated with information out of queue record (in routine NTPRCJB after having updated the due date information).

IPW$$OF: The information about completion message in the queue record is updated before writing the queue record to tape (near label SQW0). Set indication in queue record that information out of job header is not to be used to issue a completion message for the releaser, because it is too difficult to update the job header record and write the updated job header to tape.

External Device Support

The external device support interface, also referred to as Device Serving Support (DSS) is the general interface for all components residing in another partition to perform as VSE/POWER LST/PUN tasks. The VSE/POWER DSS supports any devices by providing a standard interface which allows the designers of the device to design their own specialized device support. Any IBM subsystem or component using the device serving support (DSS) is called Device Driving System (DDS).

- The external device support consists of a set of interfaces, controls and capabilities which permit moving certain functions from VSE/POWER into another partition. In this case, the functions moved are accessing the VSE/POWER spool files and driving the device.
- The external device continues to appear to the operator as a VSE/POWER controlled device. This concept is referred to in the following design as 'single system image'.

- Output scheduling remains under control of VSE/POWER.

The VSE/AF XPCC support is utilized for communication between the DSS (VSE/POWER) and the DDS.

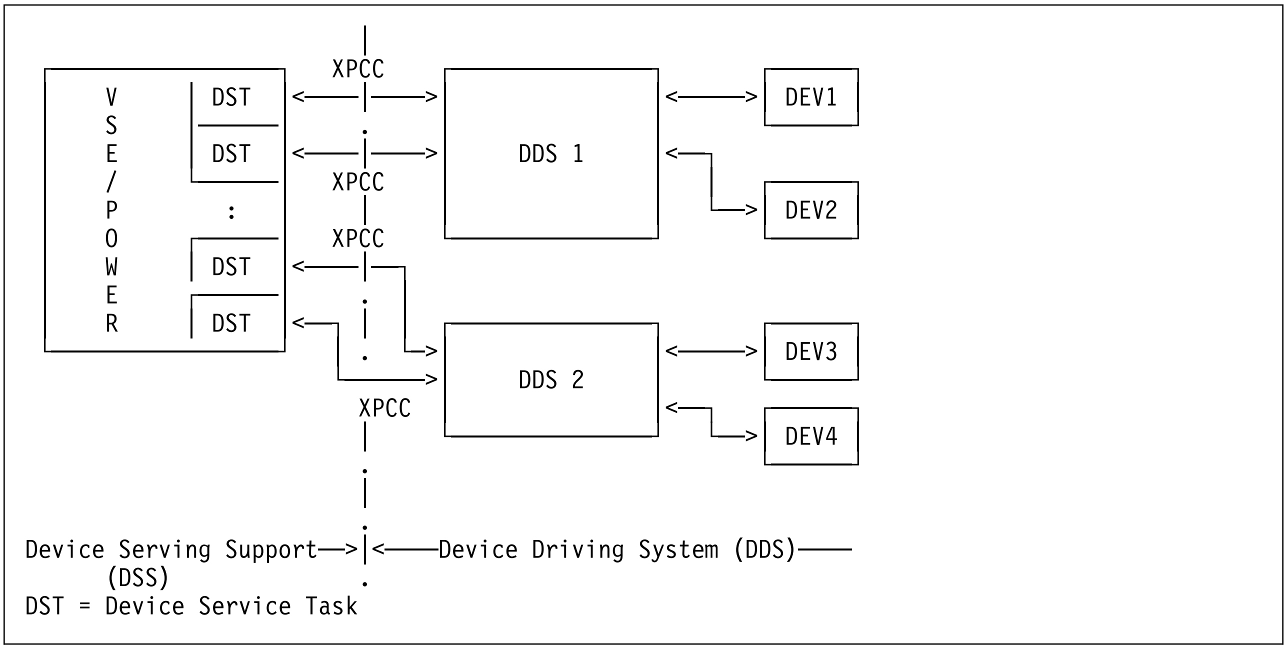


Figure 131. Device Serving Support - Device Driving System Overview

Device Service Task (DST)

The device service task performs all of the functions necessary to support a external device running under the control of a DDS in another partition. It provides the interface between the DDS and the rest of the VSE/POWER functions. It sends any order to the DDS and processes any order responses as well as the GET or CTL function of the SAS interface.

Starting an External Device: When the PSTART DEV command is entered by the operator, the VSE/POWER command processor attaches a device service task. The DST attempts first to establish a communication path to the corresponding DDS. The XPCC application id is the DDS name as specified in the PSTART command. VSE/POWER assumes that the DDS is started when a PSTART command is given by the operator and that the DDS has the 'connect any' request outstanding in order to satisfy a connection request from VSE/POWER.

The DST then issues a two-minute timer event and waits for the connection completion. If the timer expires before the communication path is established, message 1QY0I is issued to the device owning operator (person who issued the PSTART command), indicating that the DDS needed to support the device either failed or is not yet started. The message is to convey to the operator that there might be a problem with the DDS and that some sort of intervention may be required. The operator in turn may decide either to let the DST wait until the connection completes or to stop the terminal printer by means of the PSTOP DEV command.

If the operator issues a command to the terminal printer prior to the successful start of the device, the command is rejected by the VSE/POWER command processor during this phase of initialization. If, however, the command is a PSTOP DEV command, the command processor informs the DST to abandon the connection attempt.

When the connection completes, VSE/POWER expects that the DDS requests the transmission of the 'start device' order. If the first request from the DDS is not a 'return order' request, VSE/POWER discontinues the communication path by issuing the XPCC FUNC=DISCPRG macro instruction. The 'start device' order allows the DDS to perform its device specific startup process. It is the responsibility of the DDS to respond to VSE/POWER, indicating if the device is ready or not. If the device cannot be started, the DDS indicates the reason in the response order record. Following reasons are seen so far:

- Device unknown
- Device in use (busy)
- Device out of service
- Device start rejected
- Invalid parameters passed
- Start of device not accepted due to lack of resources

The 'start device' order control record contains among others PSTART command parameters in fixed format. VSE/POWER allows to specify device related information via the PARM operand on the PSTART command. The PARM operand is not checked by VSE/POWER and the unaltered command is passed to the DDS in the 'start device' order control record.

If the DDS responds saying that the device could not be started due to one of above mentioned conditions, the DST issues message 1QY1I to the command originator, telling him why the device is not started and discontinues the communication path. The DST then detaches itself after cleaning up of all acquired resources.

If the 'start device' order response is received by the DST and the DDS indicated that the device is started, message 1QY3I is displayed on the system console and also sent to the device owner. The external device control block (EDCB) is marked that the device is started. The DST is then ready to accept GET/CTL requests from the DDS. Prior to requesting the first output queue entry, the DDS can send a 'set logical destination' order to VSE/POWER. This order must be sent, when the external device should process output queue entries which are destined for a destination other than the external device name. Up to 8 logical destinations can be specified in the 'set logical destination' order, thereby overwriting the default logical destination name which is the PSTART device name; the latter must still be used to address the external device by means of the DEV commands.

```
 DDS                                 VSE/POWER

 IDENTIFY   (XPCC FUNC=IDENT)                |
  |                                          |
CONNECT any -(XPCC FUNC=CONNECT)--->         |
  |                                          |
wait (CECB)                                  |
  |          <-------------------- CONNECT with DDS-id
  |                                  wait (CECB)
  |                                wait (RECB)
get 'start device' order:                    |
 SENDR:-(return order)---------------->      |
  |                                  RECEIVE
wait (SECB)                          • pass 'start device' order
  |   <---(order control record)----  REPLY
  |                                          |
respond to order:                            |
 SENDR:-(order reponse record)-------->      |
  |                                  RECEIVE
wait (SECB)                          • process response
  |   <---------------(ok)----------  REPLY
  |                                          |
  |                                wait (RECB)
set logical destinations:                    |
 SENDR:-(set log. destination order)-->      |
  |                                  RECEIVE
wait (SECB)                          • save log. destinations
  |                                    in EDCB
  |   <---(order response record)---  REPLY
  |                                          |
get output (generic GET):                    |
 SENDR:  ---(SPL)--------------->            |
  |                                  RECEIVE
wait (SECB)                                  |
  |   <---(SPL, describing output)--  REPLY
  |                                          |
                  :
```

Figure 132. Normal Protocol to Start a Communication

Processing of Output Queue Entries: If the DDS is ready for work, it should issue a generic GET request in order to retrieve the first eligible output queue entry destined for the terminal printer. The device service task scans the specified class chains for a queue entry either in disposition 'D' or 'K' and destined for the device, which means the 'to' userid of the queue entry concerned must match one of the logical destination names assigned to the external device.

Note: The logical destination “LOCAL” will offer output entries to the DDS with either a "to" userid of “LOCAL” or which are destined for local processing.

Since the DDS is seen as a logical extension of VSE/POWER, no password checking is done, that means that all output queue entries destined for the terminal printer are passed to the DDS, regardless if a password is associated with the queue entry or not.

If no output queue entry can be selected, the device service task informs the operator via message 1QY2I, that the device is waiting for work and the GET request is returned, indicating that no queue entry is available. The DST then waits for another request from the DDS, a VSE/POWER operator command or for the arrival of an output queue entry destined for the terminal printer.

Note: A wait for order/signal request can only be given, after VSE/POWER has responded to a generic GET request, saying that no queue entry is eligible for processing.

After the DDS received the 'no queue entry available' indication it can either terminate the communication path, change the logical destination names by means of the set logical destination order, or wait until a queue entry becomes available or an order is queued. The device service task responds to the 'waiting for order/signal' request with either:

- An order control record
- A signal control record, indicating that a queue entry is placed in the local queues, ready for processing.

The DDS must react to the order according to the protocol, when an order is passed back by VSE/POWER and then redo the 'waiting for order/signal' request if necessary.

If a queue entry is added to the VSE/POWER output queues destined for a logical destination, the VSE/POWER add queue entry function routine scans the External Device Control Block (EDCB) queue for a matching device which is waiting for work for that class. If such an entry is found, the 'output arrived' signal control record is built and added at the tail of the order queue anchored to the EDCB and the DST is posted to forward the signal to the DDS. When the 'output arrived' signal is received by the DDS, the DDS can then re-issue the GET request.

Even if an 'output arrived' signal is passed to the DDS, VSE/POWER does not guarantee that an output queue entry is available when the DDS does a generic GET the next time. The queue entry might already been deleted, modified by the system operator or accessed by another DDS.

If, however, the DDS decides not to wait, it can terminate the communication path by issuing the XPCC FUNC=DISCONN or DISCPRG macro. In this case VSE/POWER informs the system operator as well as the device owner via message 1QY5I, frees all resources occupied by the DST, unchains the EDCB from the EDCB chain and detaches the DST.

If, however, an eligible queue entry is found, the DST passes information, describing the characteristics of the queue entry, such as FCB name, forms-id etc., to the DDS. The DDS may now decide if a setup is necessary or not. This, however, is the responsibility of the DDS. If a setup is required (e.g. forms or flash change), the DDS must send a 'send message' order control record to VSE/POWER. The 'send message' order contains information to whom VSE/POWER should send the message as well as the free format message itself (see “Orders from the DDS (Inbound)” on page 383 for more details on how to send messages to VSE/POWER). The DDS then waits for re-activation by the operator (PGO). VSE/POWER forwards the unaltered message to the specified operator. If requested, the DST holds a copy of the message in storage, so that the central operator can interrogate by means of the DM command why the task is operator bound. This is helpful, in the case where the 'device preparation' message does not reach its destination.

After the send message order is sent, the DDS must send a 'waiting for order/signal' request to VSE/POWER. This causes that VSE/POWER passes the first queued order or signal to the DDS. When, however, no order or signal is queued, the DST waits until an order or signal will be queued. If the returned order is not a reactive or setup device order, the DDS must reissue the 'waiting for order/signal' request until it received one of above orders.

If the operator enters the PGO DEV command, indicating that the intervention has been satisfied, the VSE/POWER command processor builds a 'reactivate device' order control record, anchors the order at the tail of the order queue of the appropriate DST and posts the DST to forward the order to the DDS. After the DDS received the 'reactivate device' order, it can continue processing. Figure 133 on page 378 shows the details about the necessary steps. The DDS can then retrieve logical records from the previously acquired queue entry using the GETRQ request of the SAS interface after it has responded to the 'reactivate device' order with an order response control record.

```
 DDS                                VSE/POWER
                  :
  |                                     |
get output (generic GET)                |
 SENDR:  ──(SPL)────────────────>       |
  |                                RECEIVE SPL
  |                                 • check SPL
wait (SECB)                         • find eligible queue entry
  |                                 • fill up SPL
  |                                     |
  |   <──(SPL, describing output)─── REPLY:
  |                                  wait (RECB)
send message order:                     |
SENDR ───────────────────────────>      |
  |                                 RECEIVE
wait (SECB)                          • process order
  |                                    - inform operator
  |   <────(order response record)─── REPLY
  |                                  wait (RECB)
  |                                     |
wait for order/signal:                  |
 SENDR:  ───(wait for order/signal)───> |
  |                                 RECEIVE
wait (SECB)                             |
                  :
                  :
  |      <────(reactivate order)───── REPLY
  |                                     |
  |                                  wait (RECB)
respond to order:                       |
 SENDR:─(order reponse record)──────>   |
  |                                 RECEIVE
wait (SECB)                         • process response
  |   <───────────────(ok)─────────  REPLY
  |                                     |
  |                                 wait (RECB)
GETRQ:                                  |
 request first/next records             |
 SENDR──────────────────────────────>   |
  |                                     |
     :
```

Figure 133. External Device Reactivation Protocol

When the operator (device owner) enters the PSETUP DEV command in order to verify the just processed printer setup or to manually adjust the forms alignment, the VSE/POWER command processor builds a 'setup device' order control record, anchors the record at the end of the order queue of the DST concerned and posts the DST to forward the order to the DDS. After the DDS has received the 'setup device' order, it can either reject the order or handle the order accordingly. If the 'setup device' order is rejected by the DDS as indicated in the order response control record, the DST issues message 1QY6I to inform the command issuer.

If the DDS accepts the 'setup device' order, it must request the first n records composing the operator wanted number of pages from VSE/POWER. This is done by using the GETRQ request of the SAS interface. Furthermore the DDS is responsible for changing all alphabetic characters to 'X' and all numeric characters to '9', if required, before printing the page on the external device. When the number of requested pages is processed, the DDS must send a 'setup processed' signal to VSE/POWER. If the DST receives such a signal, it automatically re-positions the queue entry at the beginning and waits for reactivation by the operator (device owner) by means of the PGO DEV command. The DDS must then send a 'waiting for order/signal' request in order to get the reactivate device order or any other order.

After end-of-data is detected by the DDS and the DDS has ensured that all data is printed out, the DDS notifies VSE/POWER to close the queue entry. Depending on the disposition, the queue entry is either purged from the output queue or returned to the queue with disposition 'L'.

After the DDS has processed a queue entry it can ask for new work by issuing another generic GET request.

At any time while retrieving data records for a queue entry, the DDS may 'quit' processing and free the queue entry. VSE/POWER considers the queue entry as not being processed completely and re-queues the queue entry with its original disposition for output selection by other DDSs.

Flushing an Output Queue Entry: If the operator enters the PFLUSH DEV command, requesting to flush printing of the current queue entry, the VSE/POWER command processor builds a 'flush' order control record and anchors the order at the end of the order queue associated with the DST concerned and posts the DST to forward the order. After the DDS has received the order, it can still continue printing; for example, the DDS may request more data records from VSE/POWER by means of the GETRQ request. It is the responsibility of the DDS to decide when the flush should take place. The queue entry is flushed by sending a 'close queue entry' request to VSE/POWER. After the DST has responded to the close request, the DDS can request a new output queue entry.

If flush with HOLD is requested, the DDS must continue printing until the end of the current page is reached. This in fact could require that the DDS asks for the next n records of the queue entry to be flushed. When the end of the page (or any other reasonable boundary) is reached as determined by the DDS, the DDS can take a checkpoint before flushing the queue entry.

```
  DDS                                  VSE/POWER
                     :
 return order:                         wait (RECB)
  SENDR:  ——(return order)———————>         |
   |                                   RECEIVE
   |                                       |
 wait (SECB)                               |
   |      <———(flush order)——————————  REPLY
   |                                       |
 respond to order:                         |
  SENDR:—(order reponse record)—————————>  |
   |                                   RECEIVE
 wait (SECB)                           • process response
   |    <————————————————(ok)——————————  REPLY
   |                                       |
 GETRQ:                                    |          <—————┐
  request next records                 wait (RECB)          |
  SENDR——————————————————————————————>     |          repeat until
   |                                       |          page boundary
 wait (SECB)                               |           reached
   |    <————————(data buffer)—————————  REPLY   ——————————————┘
   |                                       |
                     :
 . empty device pipeline                   |
   |                                   wait (RECB)
CHECKPOINT:                                |
 take checkpoint                           |
 SENDR ——checkpoint control record———>     |
   |                                   RECEIVE
   |                                    • perform checkpointing
 wait (SECB)                               |
   |    <——(checkpoint response rec)——  REPLY
   |                                       |
FLUSH—HOLD                                 |
 flush current queue entry                 |
 SENDR —(user data='flush—hold')—————>     |
   |                                    • re—queue queue entry
 wait (SECB)                              if applicable
   |    <——————————————————(ok)————————  REPLY
   |                                       |
                     :
```

Figure 134. Flush HOLD Protocol

Restarting an Active Output Queue Entry: If the operator enters the PRESTART DEV command, the VSE/POWER command processor builds a 'restart device' order control record and queues this record at the end of the order chain. The order is then passed to the DDS by the DST. It is the responsibility of the DDS to decide when the restart should take place. The DDS must calculate the logical record number associated with the page or line from where to begin and send a 'restart control record' to VSE/POWER. VSE/POWER then re-positions at the record specified in the restart control record and passes the next n records back to the DDS. If the specified record number is outside of the valid range, appropriate return and feedback code is setup and returned to the DDS.

Termination of an External Device: An external device is stopped when the operator enters the PSTOP DEV command or at VSE/POWER shutdown time when the operator has given the PEND command. The VSE/POWER command processor builds a 'stop device' order control record and adds it to the order chain of the appropriate DST. The DST is then posted to forward the order to the DDS. After the 'stop device' order has been sent, the DST waits for the response from the DDS. When an active external device is stopped, the DDS must quiesce its output processing by completing the current queue entry selected for the external device. This allows the DDS to drain the device pipeline. If the pipeline is emptied, the DDS can stop the external device and signal the DST, that the device is stopped. When the device stopped signal is received, the DST informs the device owner and the command issuer via message 1QY4I and terminates the communication path. The DST then detaches itself after cleaning up and releasing of all resources.

However, the operator (device owner) can request to stop the external device immediately. In this case, the DDS should purge the data already buffered in the external device, if possible, and free the queue entry. The DST re-queues the queue entry with its original disposition.

```
 DDS                                 VSE/POWER
                    :
return order:                        wait (RECB)
 SENDR:  ——(return order)—————————>      |
  |                                  RECEIVE
  |                                      |
wait (SECB)                              |
  |      <————(stop device order)————  REPLY
  |                                      |
respond to order:                        |
 SENDR:—(order reponse record)————————>  |
  |                                  RECEIVE
wait (SECB)                          • process response
  |   <———————————————(ok)——————————  REPLY
  |                                      |
finish up current queue entry            |
GETRQ:                                   |             <——————┐
 request next records                wait (RECB)              |
 SENDR———————————————————————————————>   |               repeat until
  |                                      |               end—of—data
wait (SECB)                              |                reached
  |   <————————(data buffer)—————————  REPLY                  |
  |                                      |             ———————┘
                   :
  |                                  wait (RECB)
. empty printer pipeline                 |
. stop printer                           |
  |                                      |
SENDR —(SIGNAL: 'device stopped'————>    |
  |                                   • process signal
  |                                   • inform operator
  |   <————(null buffer)————————————  REPLY
. check VSE reason code                  |
  |                                   • discontinue
DISCONNECT ———————————————————>         communication path
  |       <———————————————————————   DISCONNECT
  |                                   • detach DST
```

Figure 135. External Device Termination Protocol

When stop with RESTART is requested, the DDS must continue printing until the end of the current page or any other reasonable boundary is reached. If the boundary is reached as detected by the DDS, a checkpoint should be taken. This is done by sending a checkpoint control record to VSE/POWER. After the checkpoint is performed the DDS must return the queue entry currently being printed to the VSE/POWER spool file. The queue entry is then returned to the spool file with its original disposition so that it can be processed at a later time.

Abnormal Termination: If the DST determines that it has reached an abend situation (usually caused by a protocol violation of the DDS) it cannot recover from, it issues a XPCC FUNC=DISCPRG macro instruction at the level of failure occurred. The output queue entry currently being processed by the DDS is returned to the class chain with its original disposition.

If the DDS detects an unrecoverable error it can either abend, in which case VSE/AF notifies VSE/POWER about the situation or it can discontinue the communication path by issuing the XPCC FUNC=DISCPRG macro. In both cases, VSE/POWER informs the system operator and the device owner via message 1QY5I about the abnormal termination of the DDS, performs the necessary accounting, re-queues the queue entry being processed in the class chain and detaches the DST.

If VSE/POWER abends, the VSE/AF XPCC support informs the DDS about the abnormal termination of VSE/POWER. The DDS can empty the printer pipeline and save the last processed record number. This number can then be used by the DDS to resume processing of the queue entry from the point in the queue entry represented by the record number.

Note: Associated with each record VSE/POWER passes an internal record number. This number may be used to request restarting at that particular record.

When VSE/POWER is brought up again, all queue entries marked active are put back in the class chain with the original disposition, so that they can be processed by any other output task capable of processing them.

Checkpoint information associated with a queue entry is returned by VSE/POWER when a DDS starts accessing the queue entry (GET function). The DDS can then decide if it wants to continue from the last checkpoint, or from the record number saved by the DDS at the point when VSE/POWER abnormally terminated, or from the beginning. This can be used to resume processing of an output queue entry at a later time from the point in the output queue entry represented by the last checkpoint.

Whenever a problem occurs during output processing, the DDS may issue a “quit and lock” request for the current queue entry. This causes VSE/POWER to re-queue the queue entry with a temporary disposition of “Y” hinders any VSE/POWER task from accessing this queue entry until its disposition is changed via the PALTER command. Disposition “Y” is also forced when the XPCC communication path is discontinued and the queue entry was flagged to be held because of output processing failure, also called protect option.

Orders from VSE/POWER (Outbound)

Any operator command which either affects the DDS or needs information from the DDS is sent as an order to the DDS for processing. Each command is parsed and converted into an order. These orders are then passed to the DDS. The following orders are passed from VSE/POWER:

• Start device	(PSTART)
• Stop device	(PSTOP)
• Restart device	(PRESTART)
• Reactivate device	(PGO)
• Setup device	(PSETUP)
• Flush current processing	(PFLUSH)
• DDS defined command	(PXMIT)

Orders are processed on a 'FIFO' basis. No attempt will be done by VSE/POWER to re-arrange the orders depending on an importance level. New orders will be still accepted, even if the PSTOP DEV command was given.

Each order contains in its header section the userid and the node name of the command originator. If the command is entered by the central system operator the userid will be blank and the node name is the local node name, if one is present, or blank.

VSE/POWER requires that the DDS immediately analyses the order and answers with the order response control record, indicating if the order is accepted or what type of error condition occurred. If the order is not accepted by the DDS or the DDS indicated some other error, message 1QY6I is issued to the command submitter. The DDS might pass a message as part of the order response control record to VSE/POWER. This message is sent further to the specified person. The actual order, however, can be processed by the DDS whenever appropriate. For example, the DDS can take a checkpoint after an immediate stop order was received. The exception to above ground-rule is the 'start device' order. This order must be immediately processed by the DDS and the result returned in the order response control record.

All responses from the command, which result in a message either built by VSE/POWER or passed from the DDS with the order response control record, are forwarded to the command originator by VSE/POWER, unless overridden by the DDS.

Orders from the DDS (Inbound)

The following orders can be sent from the DDS to VSE/POWER for processing:

- Send message order
- Set logical destination order
- Advanced Function Printing (AFP) account record order

All orders from a DDS must be sent to VSE/POWER by issuing the XPCC FUNC=SENDR macro instruction. The buffer type flag in the user data of the XPCCB must indicate that the buffer contains a control record. The buffer can only contain one order control record. VSE/POWER replies to the order with the order response control record by issuing of the XPCC FUNC=REPLY macro instruction. The DDS can send an order at any point of time.

If VSE/POWER receives a 'send message' order, and the message is for the local central operator, then VSE/POWER issues the message and returns the message id to the caller. This enables the caller to issue an action message and later delete the message from the display screen using the DOM macro.

If VSE/POWER receives a 'set logical destination' order, it updates the External Device Control Block (EDCB) of the external device concerned by replacing the old logical destination names, if any, with the new ones. The new destination names are of effect when the next generic GET is done.

Heartbeat Task

The heartbeat task is used for a heartbeat connection to VSE/OCCF by means of an XPCC connection which is not used for any data transfer, but just to signal VSE/POWER that VSE/OCCF is running properly and vice versa to signal VSE/OCCF that VSE/POWER is running properly. Whenever VSE/OCCF terminates abnormally, VSE/POWER gets the appropriate return code by means of this XPCC connection. Thereupon VSE/POWER generates internally a PEND IMM command and issues the REIPL macro at the end of its own terminating process.

In order to establish the heartbeat connection, VSE/OCCF has to identify itself during the CONNECT with the application-id 'SYSOCCF' and to specify in the user field of the XPCCB the constant 'ALIVE' and a time-limit. No other data is allowed to be sent or received via this special XPCC connection. Whenever VSE/OCCF terminates abnormally, the supervisor posts this XPCC connection and VSE/POWER generates internally a PEND IMM command. This command stops immediately all activities of the VSE/POWER tasks. At the end of its termination processing, VSE/POWER issues the REIPL macro in order to automate the re-initialization of the whole system.

VSE/POWER establishes the heartbeat connection, if the XPCCB contains at connect time the following information:

IJBXTOAP (the requested application) must contain as application-id 'SYSOCCF ' (SYSOCCF padded at the right with one blank).

IJBXRUSR (the received user data) must contain the constant 'ALIVE ' (ALIVE padded at the right with one blank) and in the last two bytes the termination time-limit in minutes.

Whenever VSE/OCCF terminates abnormally, the VSE supervisor posts VSE/POWER and passes the reasoncode IJBXABDC in the field IJBXREAS. Thereupon VSE/POWER generates internally a PEND IMM command to terminate all activities. The termination time-limit specified in the last two bytes of the user data describes in minutes how long VSE/POWER has to wait before the REIPL macro can be issued in order to give the partitions controlled by VSE/POWER enough time to terminate. If the partitions do not terminate within the time-limit, a PEND FORCE command is simulated.

Once this connection has been established, VSE/POWER will never try to send any data via this connection, but just waits on the connect ECB (IJBXCECB) which gets posted if the other side issues a disconnect or terminates its processing (abnormally or normally). If the other side tries to send data, VSE/POWER will never recognize this, because VSE/POWER does not wait on the receive ECB (IJBXRECB) and never checks the appropriate fields signalling data transfer.

If the other side terminates the connection normally (e.g. by issuing a DISCPRG), VSE/POWER terminates also its connection by issuing a DISCPRG, but does not stop any other processing. On the contrary due to an always outstanding CONNECT ANY, VSE/OCCF can establish a new heartbeat task whenever it wants. Thereby it is possible to stop the unattended environment for a while to do some maintenance or testing by an operator in an 'attended' environment.

VSE/POWER may be terminating its processing due to

1. the PEND command without any operand or the PEND IMM command. In these cases VSE/POWER starts to finish its activities but keeps the heartbeat connection alive as long as possible, because it might take its time till all partitions have finished their job. At the end of VSE/POWER shutdown VSE/POWER stops the heartbeat connection by issuing a DISCPRG.
2. the PEND command with the FORCE operand or an abnormal situation. In both cases VSE/POWER does nothing concerning the heartbeat connection, but the VSE supervisor will post VSE/OCCF using the reasoncode IJBXABDC in the field IJBXREAS to indicate the failure of VSE/POWER to VSE/OCCF.

If VSE/OCCF terminates abnormally, VSE/POWER is going to shutdown the system. VSE/POWER waits according to the specified time-limit till all partitions have terminated before issuing the REIPL macro. If all partitions are unbatched before the time limit has exceeded, the REIPL macro is issued at the moment the last partition is unbatched. The end of task routine within the supervisor posts VSE/POWER every time a partition gets unbatched. If during the shutdown VSE/POWER terminates abnormally due to a program error (e.g. program check), VSE/POWER does not wait till all partitions have terminated, but issues the REIPL macro at once.

If VSE/POWER runs in an unattended environment, VSE/POWER acts differently in the following ways in order to automate the processing in an environment without operator:

1. If VSE/POWER stops a partition, the partition gets unbatched by VSE/AF. Within the partition comreg POWUNBCH is set for JCL and POWUNBTS for the end of task routine of the supervisor.
2. If VSE/POWER decides to terminate a job in a partition, VSE/POWER suppresses the dump of this partition by setting IJBNDUMP within the partition comreg.
3. If VSE/POWER tries to start a partition and this partition is not available, VSE/POWER issues message 1R68I and cancels itself. This may happen, if during the termination of VSE/POWER a partition did not terminate and VSE/OCCF tries to restart VSE/POWER. If in this case VSE/POWER cancels itself, VSE/OCCF will decide after some unsuccessful retries to re-ipl the whole system. After re-ipl all partitions will be available to be started by VSE/POWER.
4. If the heartbeat connection terminates abnormally, VSE/POWER will never provide a dump, even if VSE/POWER itself terminates abnormally.

Codes using REIPL Macro

Whenever the REIPL macro is issued, the options DEVICE=CURRENT and ACTION=ERROR are used. The symptom record has the layout of the section 3 of the symptom record used by the IDUMP macro. Sofar the following return codes are provided by the appropriate phase due to the described situation:

0001	IPW$$XH: VSE/OCCF terminated abnormally, time limit not exceeded
0002	IPW$$XH: VSE/OCCF terminated abnormally, time limit exceeded
0003	IPW$$XH: VSE/OCCF terminated abnormally, time limit was 0
0004	IPW$$AT: VSE/OCCF terminated abnormally, thereafter POWER terminated abnormally

If no dump has been produced, the constant 'NONE' is used as parameter for the REIPL macro.

The heartbeat connection is a special kind of XPCC connection which means no data can be sent in either direction. Only the connect and disconnect functions of the XPPC macro should be used and only the return codes concerning the existence of the connection have to be checked.

If VSE/OCCF tries to build a second heartbeat connection, although a heartbeat connection exists already, VSE/POWER terminates a connection by issuing the XPCC macro with the parameter FUNC=DISCPRG and passing PXPRCNOC in PXPRETCD and PXP10CAA in PXPFBKCD, which are two bytes (return and feedback code) within the user data located in IJBXSUSR of the XPCCB.

The VSE/POWER heartbeat task, serves as a kind of watchman to keep track whether the other side (VSE/OCCF) is still alive or not. The task-id is XHBT which forces the chaining of the heartbeat task into the group of the cross-partition user tasks. The heartbeat task is attached by the cross-partition master task. The heartbeat task is 'normally' detached at the end of VSE/POWER shutdown by the cross-partition master task. The heartbeat task is also detached whenever the other side terminates the communication (either normally or abnormally). The code for the heartbeat task is just within one module, the module IPW$$XH:

As soon as the heartbeat task is attached, the following is done:

1. Save termination time-limit.
2. Get virtual storage with option WAIT=YES for the symptom information, anchor the storage in the CAT and initialize the symptom information for the REIPL macro.
3. If communication not yet terminated, reset posted connect ECB.
4. Do multiple wait (IPW$WFM) on:
   a. Task ECB (POWER event).
   b. Connect ECB (XPCC event)
      or time interval ECB.

Whenever the heartbeat task gets posted, the following events are tested as exclusive events:

1. If task ECB posted (by cross-partition master task in IPW$$XM during VSE/POWER shutdown right before the XPCC macro with FUNC=TERMPRG is issued):
   a. If necessary, cancel timer interval.
   b. If XPCCB storage still available:
      1) Issue DISCPRG.
      2) Test return codes of DISCPRG and issue message if necessary.
      3) If other side of the heartbeat connection did terminate abnormally, indicate REIPL necessary.
      4) Release storage of XPCCB.
   c. Reset heartbeat connection exists.
   d. Post cross-partition master task.
   e. Detach task
2. If timer interval ECB posted:
   a. Update symptom information for REIPL macro.
   b. Indicate forced cancel.
   c. The VSE/POWER macro IPW$CNC is issued to stop VSE/POWER immediately (simulating a PEND FORCE command, the terminating message 1R99I has been issued already when processing the PEND IMM command).
3. If connect ECB of XPCC support and OCCF has terminated abnormally:
   a. Reset post bit of Connect ECB.
   b. Remove connect ECB from ECB list.
   c. Issue DISCPRG.
   d. Test return codes of DISCPRG and issue message if necessary.
   e. Release storage of XPCCB.
   f. Indicate REIPL necessary.
   g. If task ECB not yet posted by IPW$$XM (i.e. temporary command processor will still be invoked):
      1) Update ECB list:
         a) Invoke temporary command processor (IPW$ICP macro with parameters avoid authority checking and use for PEND IMM command a constant field of 72 bytes) and wait till command completed.
         b) If termination time-limit not zero:
            i. Update ECB list with timer interval ECB.
            ii. Issue timer interval with time-limit.
         c) If termination time-limit is zero:
            (If OCCF issued the DISCONNECT very fast after the CONNECT, the user data might have been destroyed. this should never happen)
            i. Update symptom information for REIPL macro.
            ii. Indicate forced cancel.
            iii. Cancel VSE/POWER using macro IPW$CNC.
   h. Wait on ECB-list till it gets posted, either at the end of the timer interval or at the end of POWER shutdown by IPW$$XM.
4. If connect ECB of XPCC support and OCCF has terminated normally:
   a. Issue DISCPRG
   b. Test return codes of DISCPRG and issue message if necessary
   c. Release storage of XPCCB
   d. Detach task.

Chapter 4. Directory

The names listed in the column "Module/Microfiche Name" are the names of the respective microfiche cards.

- Determine the type of name of any program identifier (phase, module, control section, macro, or segment).
- Determine the phase with which that name is associated.
- If the name is a linkage macro, determine the invoked phase.
- If the name is a definition macro (control block, or data block), locate the matching data area by using Figure 11 on page 36 as a reference.

A reference list of messages is also included in this chapter. It relates a message with the issuing phase.

CSECT and Control Block Name List

Usually the names appearing below can be found in a dump, e.g. the CSECT name appearing at the beginning of a phase, and some may occur more than once, e.g. the name of a control block.

Name	Type	Phase	Module Microfiche Name
ACCB	Storage descriptor of control block.		
ACIE	Storage descriptor of control block		
AQCS	CSECT	IPW$$AQ	IPW$$QM
ASCS	CSECT	IPW$$AS	IPW$$AS
ASWS	Storage descriptor of control block.		
ATCS	CSECT	IPW$$AT	IPW$$AT
BACS	CSECT	IPW$$BA	IPW$$BA
BMCS	CSECT	IPW$$BM	IPW$$BM
BRCS	CSECT	IPW$$BR	IPW$$BR
BSCS	CSECT	IPW$$BS	IPW$$BS
BWCS	CSECT	IPW$$BW	IPW$$BW
CACS	CSECT	IPW$$CA	IPW$$CA
CBCS	CSECT	IPW$$CB	IPW$$CB
CCCS	CSECT	IPW$$CC	IPW$$CC
CDCS	CSECT	IPW$$CD	IPW$$CD
CECS	CSECT	IPW$$CE	IPW$$CE
CFCS	CSECT	IPW$$CF	IPW$$CF
CGCS	CSECT	IPW$$CG	IPW$$CG
CHCS	CSECT	IPW$$CH	IPW$$CH
CIB	Storage descriptor of control block		
CICS	CSECT	IPW$$CI	IPW$$CI
CIE	Storage descriptor of control block		
CI2	Storage descriptor of control block		
CJCS	CSECT	IPW$$CJ	IPW$$CJ
CLCS	CSECT	IPW$$CL	IPW$$CL
CLDCS	CSECT	IPW$$CLD	IPW$$CLD
CMCS	CSECT	IPW$$CM	IPW$$CM
CNCS	CSECT	IPW$$CN	IPW$$CN
COCB	Storage descriptor of control block		
COCS	CSECT	IPW$$CO	IPW$$CO
CPCS	CSECT	IPW$$CP	IPW$$CP
CPFCS	CSECT	IPW$$CPF	IPW$$CPF
CPSCS	CSECT	IPW$$CPS	IPW$$CPS
CRCS	CSECT	IPW$$CR	IPW$$CR
CRECS	CSECT	IPW$$CRE	IPW$$CRE
CSCS	CSECT	IPW$$CS	IPW$$CS
CSGCS	CSECT	IPW$$CSG	IPW$$CSG
CTCS	CSECT	IPW$$CT	IPW$$CT
CUCS	CSECT	IPW$$CU	IPW$$CU
CVCS	CSECT	IPW$$CV	IPW$$CV
CXCS	CSECT	IPW$$CX	IPW$$CX
CYCS	CSECT	IPW$$CY	IPW$$CY
CAT	Storage descriptor of control block.		
CIB	Storage descriptor of control block.		
COCB	Storage descriptor of control block.		
CPB	Storage descriptor of control block.		
DDCS	CSECT	IPW$$DD	IPW$$DD
DMB	Storage descriptor of control block.		

```
DPCB       Storage descriptor of control block.
DPCS       CSECT      IPW$$DP    IPW$$DP
DQCS       CSECT      IPW$$DQ    IPW$$QM
DSCS       CSECT      IPW$$DS    IPW$$DM
DTCS       CSECT      IPW$$DT    IPW$$DT
EDCB       Storage descriptor of control block.
ERCS       CSECT      IPW$$ER    IPW$$ER
FCBCB      Storage descriptor of control block.
FQCS       CSECT      IPW$$FQ    IPW$$QM
GACS       CSECT      IPW$$GA    IPW$$AM
GDCS       CSECT      IPW$$GD    IPW$$DM
GFCS       CSECT      IPW$$GF    IPW$$GF
GNCB       Storage descriptor of control block.
IBCS       CSECT      IPW$$IB    IPW$$IB
ICCS       CSECT      IPW$$IC    IPW$$CM
INCS       CSECT      IPW$$IN    IPW$$IN
IPCS       CSECT      IPW$$IP    IPW$$IP
IDCS       CSECT      IPW$$ID    IPW$$ID
I1CS       CSECT      IPW$$I1    IPW$$I1
I2CS       CSECT      IPW$$I2    IPW$$I2
I3CS       CSECT      IPW$$I3    IPW$$I3
I4CS       CSECT      IPW$$I4    IPW$$I4
I5CS       CSECT      IPW$$I5    IPW$$I5
I7CS       CSECT      IPW$$I7    IPW$$I7
JCA        Storage descriptor of control block.
LDCS       CSECT      IPW$$LD    IPW$$LD
LDCS1      CSECT      IPW$$LD1   IPW$$LD1
LDCS2      CSECT      IPW$$LD2   IPW$$LD2
LDCS3      CSECT      IPW$$LD3   IPW$$LD3
LDCS4      CSECT      IPW$$LD4   IPW$$LD4
LDCS5      CSECT      IPW$$LD5   IPW$$LD5
LFCS       CSECT      IPW$$LF    IPW$$LF
LHCS       CSECT      IPW$$LH    IPW$$LH
LMCS       CSECT      IPW$$LM    IPW$$LM
LOCS       CSECT      IPW$$LO    IPW$$LO
LNCS       CSECT      IPW$$SN    IPW$$LN
LRCB       Storage descriptor of control block.
LRCS       CSECT      IPW$$LR    IPW$$LR
LUCB       Storage descriptor of control block.
LUCS       CSECT      IPW$$LU    IPW$$LU
LWCS       CSECT      IPW$$LW    IPW$$LW
MCB        Storage descriptor of control block.
MDCS       CSECT      IPW$$MD    IPW$$MD
MECB       Storage descriptor of control block.
MMCS       CSECT      IPW$$MM    IPW$$MM
MMB        Storage descriptor of control block.
MPCS       CSECT      IPW$$MP    IPW$$MP
MSCB       Storage descriptor of control block.
MSCS       CSECT      IPW$$MS    IPW$$MS
MXCS       CSECT      IPW$$MX    IPW$$MX
NCB        Storage descriptor of control block.
NCCS       CSECT      IPW$$NC    IPW$$NC
NDT        Storage descriptor of control block.
NKCS       CSECT      IPW$$NK    IPW$$NK
NMCS       CSECT      IPW$$NM    IPW$$NM
NPCS       CSECT      IPW$$NP    IPW$$NP
NQCS       CSECT      IPW$$NQ    IPW$$QM
NRCS       CSECT      IPW$$NR    IPW$$NR
```

```
NR2CS      CSECT      IPW$$NR2   IPW$$NR2
NSCS       CSECT      IPW$$NS    IPW$$NS
NTCS       CSECT      IPW$$NT    IPW$$NT
OBCS       CSECT      IPW$$OB    IPW$$OB
OCCS       CSECT      IPW$$OC    IPW$$OC
OECS       CSECT      IPW$$OE    IPW$$OE
OFCS       CSECT      IPW$$OF    IPW$$OF
OPCS       CSECT      IPW$$OP    IPW$$OP
OTCS       CSECT      IPW$$OT    IPW$$OT
PACS       CSECT      IPW$$PA    IPW$$AM
PCCS       CSECT      IPW$$PC    IPW$$PC
PDB        Storage descriptor of control block.
PDCS       CSECT      IPW$$PD    IPW$$DM
PFCS       CSECT      IPW$$PF    IPW$$PF
PNCB       Storage descriptor of control block.
PLCS       CSECT      IPW$$PL    IPW$$PL
PPCS       CSECT      IPW$$PP    IPW$$PP
PRCS       CSECT      IPW$$PR    IPW$$PR
PSCS       CSECT      IPW$$PS    IPW$$PS
PS1CS      CSECT      IPW$$PS1   IPW$$PS1
Q1CS       CSECT      IPW$$Q1    IPW$$QM
RMCB       Storage descriptor of control block.
RQCS       CSECT      IPW$$RQ    IPW$$QM
RYCS       CSECT      IPW$$RY    IPW$$RY
SACS       CSECT      IPW$$SA    IPW$$AM
SCB        Storage descriptor of control block.
SSCB       Storage descriptor of control block.
SCCS       CSECT      IPW$$SC    IPW$$SC
SDCB       Storage descriptor of control block.
SDCS       CSECT      IPW$$SD    IPW$$SD
SER        Storage descriptor of control block.
SFCS       CSECT      IPW$$SF    IPW$$SF
SLCS       CSECT      IPW$$SL    IPW$$SL
SMCS       CSECT      IPW$$SM    IPW$$SM
SNCB       Storage descriptor of control block.
SNCS       CSECT      IPW$$SN    IPW$$SN
SPB        Storage descriptor of control block.
SQCS       CSECT      IPW$$SQ    IPW$$QM
SRCS       CSECT      IPW$$SR    IPW$$SR
SRQE       Storage descriptor of control block.
SSCS       CSECT      IPW$$SS    IPW$$SS
SUCB       Storage descriptor of control block.
SXCS       CSECT      IPW$$SX    IPW$$SX
SYCS       CSECT      IPW$$SY    IPW$$SY
S1CS       CSECT      IPW$$S1    IPW$$S1
S2CS       CSECT      IPW$$S2    IPW$$S2
S3CS       CSECT      IPW$$S3    IPW$$S3
TBB        Storage descriptor of control block.
TCB        Storage descriptor of control block.
TCCS       CSECT      IPW$$TC    IPW$$TC
TDCB       Storage descriptor of control block.
TDCS       CSECT      IPW$$TD    IPW$$TD
TSCS       CSECT      IPW$$TS    IPW$$TS
TIB        Storage descriptor of control block.
TICS       CSECT      IPW$$TI    IPW$$TI
TQCS       CSECT      IPW$$TQ    IPW$$TQ
TRCS       CSECT      IPW$$TR    IPW$$TR
TSCS       CSECT      IPW$$TS    IPW$$TI
```

```
TVCS       CSECT      IPW$$TV    IPW$$TV
T1CS       CSECT      IPW$$T1    IPW$$T1
VDCB       Storage descriptor of control block.
VECS       CSECT      IPW$$VE    IPW$$VE
VSB        Storage descriptor of control block.
VSCB       Storage descriptor of control block.
WACB       Storage descriptor of control block.
WCB        Storage descriptor of control block.
XHCS       CSECT      IPW$$XH    IPW$$XH
XMCS       CSECT      IPW$$XM    IPW$$XM
XJCS       CSECT      IPW$$XJ    IPW$$XJ
XRECS      CSECT      IPW$$XRE   IPW$$XRE
XTCS       CSECT      IPW$$XT    IPW$$XT
XTCCS      CSECT      IPW$$XTC   IPW$$XTC
XTGCS      CSECT      IPW$$XTG   IPW$$XTG
XTMCS      CSECT      IPW$$XTM   IPW$$XTM
XTPCS      CSECT      IPW$$XTP   IPW$$XTP
XTSCS      CSECT      IPW$$XTS   IPW$$XTS
XTWALLAR   Storage descriptor of control block.
XTWAREA    Storage descriptor of control block.
XTWFUNAR   Storage descriptor of control block.
XTWSUBAR   Storage descriptor of control block.
XWECS      CSECT      IPW$$XWE   IPW$$XWE
```

PHASE Name List

Name	Type	Phase	Module Microfiche Name
IPW$$AM	MODULE		
IPW$$AQ	PHASE	IPW$$AQ	IPW$$QM
IPW$$AS	PHASE	IPW$$AS	IPW$$DM
IPW$$AT	PHASE	IPW$$AT	IPW$$AT
IPW$$BA	PHASE	IPW$$BA	IPW$$BA
IPW$$BM	PHASE	IPW$$BM	IPW$$BM
IPW$$BR	PHASE	IPW$$BR	IPW$$BR
IPW$$BS	PHASE	IPW$$BS	IPW$$BS
IPW$$BW	PHASE	IPW$$BW	IPW$$BW
IPW$$CA	PHASE	IPW$$CA	IPW$$CA
IPW$$CAC	PHASE	IPW$$CAC	IPW$$CAC
IPW$$CB	PHASE	IPW$$CB	IPW$$CB
IPW$$CC	PHASE	IPW$$CC	IPW$$CC
IPW$$CD	PHASE	IPW$$CD	IPW$$CD
IPW$$CE	PHASE	IPW$$CE	IPW$$CE
IPW$$CF	PHASE	IPW$$CF	IPW$$CF
IPW$$CG	PHASE	IPW$$CG	IPW$$CG
IPW$$CH	PHASE	IPW$$CH	IPW$$CH
IPW$$CI	PHASE	IPW$$CI	IPW$$CI
IPW$$CJ	PHASE	IPW$$CJ	IPW$$CJ
IPW$$CL	PHASE	IPW$$CL	IPW$$CL
IPW$$CLD	PHASE	IPW$$CLD	IPW$$CLD
IPW$$CM	PHASE	IPW$$CM	IPW$$CM
IPW$$CN	PHASE	IPW$$CN	IPW$$CN
IPW$$CO	PHASE	IPW$$CO	IPW$$CO
IPW$$CP	PHASE	IPW$$CP	IPW$$CP
IPW$$CPF	PHASE	IPW$$CPF	IPW$$CPF
IPW$$CPS	PHASE	IPW$$CPS	IPW$$CPS
IPW$$CR	PHASE	IPW$$CR	IPW$$CR
IPW$$CRE	PHASE	IPW$$CRE	IPW$$CRE
IPW$$CS	PHASE	IPW$$CS	IPW$$CS
IPW$$CSG	PHASE	IPW$$CSG	IPW$$CSG
IPW$$CT	PHASE	IPW$$CT	IPW$$CT
IPW$$CU	PHASE	IPW$$CU	IPW$$CU
IPW$$CV	PHASE	IPW$$CV	IPW$$CV
IPW$$CX	PHASE	IPW$$CX	IPW$$CX
IPW$$CY	PHASE	IPW$$CSG	IPW$$CSG
IPW$$DD	PHASE	IPW$$DD	IPW$$DD
IPW$$DM	MODULE		
IPW$$DP	PHASE	IPW$$DP	IPW$$DP
IPW$$DQ	PHASE	IPW$$DQ	IPW$$QM
IPW$$DS	PHASE	IPW$$DS	IPW$$DM
IPW$$DT	PHASE	IPW$$DT	IPW$$DT
IPW$$ER	PHASE	IPW$$ER	IPW$$ER
IPW$$FQ	PHASE	IPW$$FQ	IPW$$QM
IPW$$GA	PHASE	IPW$$GA	IPW$$AM
IPW$$GD	PHASE	IPW$$GD	IPW$$DM
IPW$$GF	PHASE	IPW$$GF	IPW$$GF
IPW$$IB	PHASE	IPW$$IB	IPW$$IB
IPW$$IC	PHASE	IPW$$IC	IPW$$CM
IPW$$ID	PHASE	IPW$$ID	IPW$$ID

IPW$$IP	PHASE	IPW$$IP	IPW$$IP
IPW$$I1	PHASE	IPW$$I1	IPW$$I1
IPW$$I2	PHASE	IPW$$I2	IPW$$I2
IPW$$I3	PHASE	IPW$$I3	IPW$$I3
IPW$$I4	PHASE	IPW$$I4	IPW$$I4
IPW$$I5	PHASE	IPW$$I5	IPW$$I5
IPW$$I7	PHASE	IPW$$I7	IPW$$I7
IPW$$IN	PHASE	IPW$$IN	IPW$$IN
IPW$$LD	PHASE	IPW$$LD	IPW$$LD
IPW$$LD1	PHASE	IPW$$LD1	IPW$$LD1
IPW$$LD2	PHASE	IPW$$LD2	IPW$$LD2
IPW$$LD3	PHASE	IPW$$LD3	IPW$$LD3
IPW$$LD4	PHASE	IPW$$LD4	IPW$$LD4
IPW$$LD5	PHASE	IPW$$LD5	IPW$$LD5
IPW$$LF	PHASE	IPW$$LF	IPW$$LF
IPW$$LH	PHASE	IPW$$LH	IPW$$LH
IPW$$LM	PHASE	IPW$$LM	IPW$$LM
IPW$$LN	PHASE	IPW$$LN	IPW$$LN
IPW$$LO	PHASE	IPW$$LO	IPW$$LO
IPW$$LR	PHASE	IPW$$LR	IPW$$LR
IPW$$LU	PHASE	IPW$$LU	IPW$$LU
IPW$$LW	PHASE	IPW$$LW	IPW$$LW
IPW$$MM	PHASE	IPW$$MM	IPW$$MM
IPW$$MP	PHASE	IPW$$MP	IPW$$MP
IPW$$MS	PHASE	IPW$$MS	IPW$$MS
IPW$$MX	PHASE	IPW$$MX	IPW$$MX
IPW$$NC	PHASE	IPW$$NC	IPW$$NC
IPW$$NK	PHASE	IPW$$NK	IPW$$NK
IPW$$NM	PHASE	IPW$$NM	IPW$$NM
IPW$$NP	PHASE	IPW$$NP	IPW$$NP
IPW$$NQ	PHASE	IPW$$NQ	IPW$$QM
IPW$$NU	PHASE	IPW$$NU	IPW$$NU
IPW$$NR	PHASE	IPW$$NR	IPW$$NR
IPW$$NR2	PHASE	IPW$$NR2	IPW$$NR2
IPW$$NS	PHASE	IPW$$NS	IPW$$NS
IPW$$NT	PHASE	IPW$$NT	IPW$$NT
IPW$$OB	PHASE	IPW$$OB	IPW$$OB
IPW$$OC	PHASE	IPW$$OC	IPW$$OC
IPW$$OE	PHASE	IPW$$OE	IPW$$OE
IPW$$OF	PHASE	IPW$$OF	IPW$$OF
IPW$$OT	PHASE	IPW$$OT	IPW$$OT
IPW$$PA	PHASE	IPW$$PA	IPW$$AM
IPW$$PC	PHASE	IPW$$PC	IPW$$PC
IPW$$PD	PHASE	IPW$$PD	IPW$$DM
IPW$$PF	PHASE	IPW$$PF	IPW$$PF
IPW$$PL	PHASE	IPW$$PL	IPW$$PL
IPW$$PP	PHASE	IPW$$PP	IPW$$PP
IPW$$PR	PHASE	IPW$$PR	IPW$$PR
IPW$$PS	PHASE	IPW$$PS	IPW$$PS
IPW$$PS1	PHASE	IPW$$PS1	IPW$$PS1
IPW$$OP	PHASE	IPW$$OP	IPW$$OP
IPW$$QM	MODULE		
IPW$$Q1	PHASE	IPW$$Q1	IPW$$QM
IPW$$RQ	PHASE	IPW$$RQ	IPW$$QM
IPW$$RY	PHASE	IPW$$RY	IPW$$RY
IPW$$SA	PHASE	IPW$$SA	IPW$$AM
IPW$$SC	PHASE	IPW$$SC	IPW$$SC
IPW$$SD	PHASE	IPW$$SD	IPW$$SD

```
IPW$$SE    PHASE      IPW$$SE    IPW$$SE
IPW$$SF    PHASE      IPW$$SF    IPW$$SF
IPW$$SL    PHASE      IPW$$SL    IPW$$SL
IPW$$SM    PHASE      IPW$$SM    IPW$$SM
IPW$$SN    PHASE      IPW$$SN    IPW$$SN
IPW$$SQ    PHASE      IPW$$SQ    IPW$$QM
IPW$$SR    PHASE      IPW$$SR    IPW$$SR
IPW$$SS    PHASE      IPW$$SS    IPW$$SS
IPW$$SY    PHASE      IPW$$SY    IPW$$SY
IPW$$S1    PHASE      IPW$$S1    IPW$$S1
IPW$$S2    PHASE      IPW$$S2    IPW$$S2
IPW$$S3    PHASE      IPW$$S3    IPW$$S3
IPW$$TC    PHASE      IPW$$TC    IPW$$TC
IPW$$TD    PHASE      IPW$$TD    IPW$$TD
IPW$$TI    PHASE      IPW$$TI    IPW$$TI
IPW$$TQ    PHASE      IPW$$TQ    IPW$$TQ
IPW$$TR    PHASE      IPW$$TR    IPW$$TR
IPW$$TS    PHASE      IPW$$TS    IPW$$TS
IPW$$TV    PHASE      IPW$$TV    IPW$$TV
IPW$$T1    PHASE      IPW$$T1    IPW$$T1
IPW$$VE    PHASE      IPW$$VE    IPW$$VE
IPW$$XH    PHASE      IPW$$XH    IPW$$XH
IPW$$XJ    PHASE      IPW$$XJ    IPW$$XJ
IPW$$XM    PHASE      IPW$$XM    IPW$$XM
IPW$$XRE   PHASE      IPW$$XRE   IPW$$XRE
IPW$$XT    PHASE      IPW$$XT    IPW$$XT
IPW$$XTC   PHASE      IPW$$XTC   IPW$$XTC
IPW$$XTG   PHASE      IPW$$XTG   IPW$$XTG
IPW$$XTM   PHASE      IPW$$XTM   IPW$$XTM
IPW$$XTP   PHASE      IPW$$XTP   IPW$$XTP
IPW$$XTS   PHASE      IPW$$XTS   IPW$$XTS
IPW$$XWE   PHASE      IPW$$XWE   IPW$$XWE
```

Macro List

Macro	Type	Phase
CTLSPOOL	LINKAGE	IPW$$SM
GETSPOOL	LINKAGE	IPW$$SM
IPW$AJ#	SERVICE	
IPW$AQS	LINKAGE	IPW$$AQ
IPW$ATT	LINKAGE	IPW$$NU
IPW$BUF	LINKAGE	IPW$$BS
IPW$CAF	LINKAGE	IPW$$GA
IPW$CLI	LINKAGE	Note 1
IPW$CNC	LINKAGE	
IPW$CPY	COPYRIGHT	
IPW$CTT	LINKAGE	IPW$$NU
IPW$DAB	DEFINITION	
IPW$DAC	DEFINITION	
IPW$DBA	DEFINITION	
IPW$DBC	DEFINITION	
IPW$DCB	DEFINITION	
IPW$DCI	DEFINITION	
IPW$DCM	DEFINITION	
IPW$DCO	DEFINITION	
IPW$DCP	DEFINITION	
IPW$DCM	DEFINITION	
IPW$DCW	DEFINITION	
IPW$DCT	DEFINITION	
IPW$DDE	DEFINITION	
IPW$DDR	DEFINITION	
IPW$DED	DEFINITION	
IPW$DEF	DEFINITION	
IPW$DET	LINKAGE	IPW$$NU
IPW$DFC	DEFINITION	
IPW$DGN	DEFINITION	
IPW$DJK	DEFINITION	
IPW$DKA	DEFINITION	
IPW$DLC	DEFINITION	
IPW$DLR	DEFINITION	
IPW$DLU	DEFINITION	
IPW$DLW	DEFINITION	
IPW$DMC	DEFINITION	
IPW$DMD	DEFINITION	
IPW$DMM	DEFINITION	
IPW$DMS	DEFINITION	
IPW$DNC	DEFINITION	
IPW$DNR	DEFINITION	
IPW$DOP	DEFINITION	
IPW$DPA	DEFINITION	
IPW$DPD	DEFINITION	
IPW$DPN	DEFINITION	
IPW$DPW	DEFINITION	
IPW$DQC	DEFINITION	
IPW$DQR	DEFINITION	
IPW$DQS	LINKAGE	IPW$$DQ
IPW$DRM	DEFINITION	
IPW$DRQ	DEFINITION	
IPW$DSA	DEFINITION	

IPW$DSC	DEFINITION	
IPW$DSD	DEFINITION	
IPW$DSL	DEFINITION	
IPW$DSN	DEFINITION	
IPW$DSP	DEFINITION	
IPW$DSR	DEFINITION	
IPW$DSS	DEFINITION	
IPW$DSU	DEFINITION	
IPW$DSV	DEFINITION	
IPW$DTB	DEFINITION	
IPW$DTC	DEFINITION	
IPW$DTE	DEFINITION	
IPW$DTI	DEFINITION	
IPW$DTX	DEFINITION	
IPW$DVC	DEFINITION	
IPW$DVD	DEFINITION	
IPW$DVP	DEFINITION	
IPW$DVS	DEFINITION	
IPW$DWA	DEFINITION	
IPW$DWC	DEFINITION	
IPW$DWG	DEFINITION	
IPW$DWN	DEFINITION	
IPW$DWP	DEFINITION	
IPW$DXE	DEFINITION	
IPW$DXW	DEFINITION	
IPW$EQU	DEFINITION	
IPW$FQS	LINKAGE	IPW$$FQ
IPW$GAM	LINKAGE	IPW$$NU
IPW$GAR	LINKAGE	IPW$$GA/GF
IPW$GDR	LINKAGE	IPW$$GD
IPW$GLR	LINKAGE	Note 1
IPW$GMD	DEFINITION	
IPW$GMM	DEFINITION	
IPW$GMS	LINKAGE	IPW$$MS
IPW$GQR	LINKAGE	IPW$$NU
IPW$GQS	LINKAGE	IPW$$NQ
IPW$GSL	LINKAGE	IPW$$SL
IPW$GTE	SERVICE	IPW$$NU
IPW$GTO	LINKAGE	IPW$$TS
IPW$GTS	LINKAGE	IPW$$SS
IPW$IAS	LINKAGE	IPW$$AS
IPW$ICP	LINKAGE	IPW$$IC
IPW$ICS	SERVICE	IPW$$NU
IPW$IDM	LINKAGE	IPW$$ID
IPW$IDS	LINKAGE	IPW$$DS
IPW$IIS	LINKAGE	IPW$$PS1
IPW$ITP	LINKAGE	IPW$$TS
IPW$IOC	LINKAGE	IPW$$OB
IPW$IOM	LINKAGE	IPW$$BM/NM/SR
IPW$IOR	DEFINITION	
IPW$IQS	LINKAGE	IPW$$SQ
IPW$IPS	LINKAGE	IPW$$LD1/LD2/LD3/LD4/LD5
IPW$IRY	LINKAGE	IPW$$RY
IPW$ITQ	LINKAGE	IPW$$TQ
IPW$ITS	LINKAGE	IPW$$SS
IPW$IXS	LINKAGE	IPW$$XTS
IPW$MXD	DEFINITION	
IPW$MQR	LINKAGE	IPW$$NU

IPW$NTY	SERVICE	IPW$$NU
IPW$OAF	LINKAGE	IPW$$GA/GF
IPW$OEF	LINKAGE	IPW$$OE
IPW$OLI	LINKAGE	Note 1
IPW$OPI	LINKAGE	IPW$$OP
IPW$OTP	LINKAGE	IPW$$OT
IPW$PAR	LINKAGE	IPW$$PA/PF
IPW$PDR	LINKAGE	IPW$$PD
IPW$PLR	LINKAGE	Note 1
IPW$RDC	LINKAGE	IPW$$NU
IPW$RDD	LINKAGE	IPW$$NU
IPW$RDQ	LINKAGE	IPW$$NU
IPW$RDT	LINKAGE	IPW$$NU
IPW$RET	LINKAGE	Note 2
IPW$RLR	LINKAGE	IPW$$NU
IPW$RLV	LINKAGE	IPW$$NU
IPW$RLW	LINKAGE	IPW$$NU
IPW$RMS	LINKAGE	IPW$$NU
IPW$RQS	LINKAGE	IPW$$RQ
IPW$RSR	LINKAGE	IPW$$NU
IPW$RSV	LINKAGE	IPW$$NU
IPW$RSW	LINKAGE	IPW$$NU
IPW$SAV	LINKAGE	Note 2
IPW$SRJ	LINKAGE	IPW$$SC
IPW$SSJ	LINKAGE	IPW$$PC
IPW$STM	LINKAGE	IPW$$NU
IPW$SXJ	LINKAGE	IPW$$XJ
IPW$TDM	SERVICE	IPW$$NU
IPW$TTM	LINKAGE	IPW$$TS
IPW$TTS	LINKAGE	IPW$$SS
IPW$TRC	DEF+LINKAGE	IPW$$NU
IPW$ULP	LINKAGE	IPW$$LU
IPW$UNV	LINKAGE	IPW$$NU
IPW$VCA	LINKAGE	IPW$$CM
IPW$VDA	LINKAGE	IPW$$NU
IPW$WFB	LINKAGE	
IPW$WFC	LINKAGE	IPW$$NU
IPW$WFD	LINKAGE	IPW$$NU
IPW$WFE	LINKAGE	IPW$$NU
IPW$WFI	LINKAGE	IPW$$NU
IPW$WFL	LINKAGE	IPW$$NU
IPW$WFM	LINKAGE	IPW$$NU
IPW$WFO	LINKAGE	IPW$$NU
IPW$WFQ	LINKAGE	IPW$$NU
IPW$WFS	LINKAGE	IPW$$NU
IPW$WFX	LINKAGE	IPW$$NU
IPW$WQR	LINKAGE	IPW$$NU
IPW$WTD	LINKAGE	IPW$$NU
IPW$WTO	LINKAGE	IPW$$NU
IPW$WTQ	LINKAGE	IPW$$NU
IPW$WTR	LINKAGE	IPW$$NU
IPW$WTT	LINKAGE	IPW$$NU
IPWSEGM	LINKAGE	IPW$$NU
PACCNT	GENERATION	
PCPTAB	GENERATION	
PLINE	GENERATION	
PNODE	GENERATION	(User-specified phase name)
POWER	GENERATION	(User-specified phase name)

PRMT	GENERATION	
PUTACCT	LINKAGE	IPW$$NU
PUTSPOOL	LINKAGE	IPW$$SM
PWRSPL	DEFINITION	
SEGMENT	LINKAGE	IPW$$NU
SPL	DEFINITION	

Notes:

1. Refer to “Interface Linkage” on page 21.
2. Refer to “Function Linkage” on page 21. For linkage conventions and register saving conventions, refer to the appropriate sections of the TCB, which is described in Chapter 5, “Storage Layout and Data Areas” on page 433 of this book.

Macro Shipables' List

This list indicates the macros that will be shipped and whether optional or not.

Macro	Type	Note
CTLSPOOL	LINKAGE	Required
GETSPOOL	LINKAGE	Required
IPW$CPY	COPYRIGHT	Optional
IPW$DDE	DEFINITION	Optional
IPW$DEF	DEFINITION	Optional
IPW$DLW	DEFINITION	Optional
IPW$DNC	DEFINITION	Optional
IPW$DNR	DEFINITION	Optional
IPW$DPA	DEFINITION	Optional
IPW$DPD	DEFINITION	Optional
IPW$DPN	DEFINITION	Optional
IPW$DQC	DEFINITION	Optional
IPW$DQR	DEFINITION	Optional
IPW$DSD	DEFINITION	Optional
IPW$DTC	DEFINITION	Optional
IPW$DTX	DEFINITION	Optional
IPW$DXE	DEFINITION	Required
IPW$EQU	DEFINITION	Optional
IPW$GQR	LINKAGE	Optional
IPW$IDM	LINKAGE	Required
IPW$MXD	DEFINITION	Required
IPW$NTY	SERVICE	Optional
IPW$RLV	LINKAGE	Optional
IPW$RSV	LINKAGE	Optional
IPW$TDM	SERVICE	Optional
IPW$TRC	DEF+LINKAGE	Optional
IPW$WTO	LINKAGE	Optional
IPWSEGM	LINKAGE	Required
PACCNT	GENERATION	Required
PCPTAB	GENERATION	Required
PLINE	GENERATION	Required
PNODE	GENERATION	Required
POWER	GENERATION	Required
PRMT	GENERATION	Required
PUTACCT	LINKAGE	Required
PUTSPOOL	LINKAGE	Required
PUTSPOOL	LINKAGE	Required
PWRSPL	DEFINITION	Required
SEGMENT	LINKAGE	Required
SPL	DEFINITION	Required

Programming Example Shipables' List

This list indicates the programming examples that will be shipped.

Example	Location
JOBEXAMP.A	PRD1.MACLIB
NETEXAMP.A	PRD1.MACLIB
OUTEXAMP.A	PRD1.MACLIB
XMTEXAMP.A	PRD1.MACLIB
PWREXAMP.A	PRD1.MACLIB
GCMEXAPM.Z	IJSYSRS.SYSLIB

Message Reference

See also “Message Service” on page 127 and “Message Handler Overview” on page 143.

Refer to “Message Coding and Documentation Considerations” on page 153 for considerations in coding VSE/POWER messages.

This chapter describes individual VSE/POWER messages and rules for their coding:

- routing code (*)
- descriptor code (*)
- VSE message color (*)
- whether command response message (*)
- whether message is DOM'ed (deleted from screen via DOM macro) (*)
- issuing macro
- message text, including indicators for:
  – message number
  – message equate suffix
  – whether message is to central operator, RJE or both
  – whether message is locally defined in module
- issuing module

(*) - central operator messages only.

The routing and descriptor codes can be found in Figure 55 on page 145 and Figure 56 on page 146.

The routing and descriptor codes are either:

- explicitly coded using the message definition macro IPW$GDM operands RT= or DC=, in the macro IPW$GMM which contains most VSE/POWER message definitions, for issuing/fetching the message via IPW$GAM, or
- set in the task TCB fields (TCMRT and TCMDC respectively) prior to issuing the IPW$WTO macro.

Message Routing Code: The developer should consider the following when deciding on how to code a message routing code (for an overview of the routing codes see Figure 55 on page 145):

Explicit Routing Code Needed:

1. when the default is inadequate, or the message should be routed:
   - to the central operator besides the Command origin (Command Response message)
   - to the central operator besides the User console (User Job message)
     e.g. It is important to specify a routing code for short-on-resource and error messages that can occur during processing of:
     – AR Command response
     – User job (Execution Reader and Writer) since without a routing code other than the default console (i.e. origin AR console or the programmer/User console) the message may not be routed to the central operator (unless the origin happens to be the central operator).
       MI (2 = Master Console Information) (RT=MI)
       SP (10 = System Programmer/Error/Maintenance) (RT=SP)
2. for special Consoles:
   SE (9 = Security Console) (RT=SE)
   TA (3 = Tape Pool) if not POFFLOAD (RT=TA)
   is needed only for messages for which the tape operator has a "need to know", i.e. 1Q55A,1Q56I,1Q57A,1Q5CI,1QB9A.
   Examples of messages not needed by tape operator:

i.e. 1R35I, 1R41D, 1R41I, 1Q5CI(1).

3. negative routing: when routing is **not** to occur to a console (NRT=): indicated by minus (-) routing, e.g. "-PG".

Routing Codes Set Automatically by the following routines:

1. VSE/POWER local message module interface IPW$$MS (CAMS+24):
   - for decision message i.e. reply messages (IPW$WTR):
     MA (1 = Master Console Action) indicated by "(MA)"
   - all AR Command Response messages will be routed to origin console (routing to origin console indicated by "(or)")
   - all messages **EXCEPT** if message is a AR command response or reply/decision msg:
     MI (2 = Master Console Information) indicated by "(MI)"

   Note: - it is important that all messages also go the central operator, e.g. 1Q51I (Execution JECL Error) causes display of bad JECL + 1R33D CORRECT JECL, which have **ALL** to be displayed at the Master console at execution time since the System operator must see all mesaages in order to correct JECL. Otherwise only the User console (ECHO=) might see the messages.

   a. if DC=DA (decision message) then RT=MA (route to Master Action console) is set
   b. if RT=MA (Master Action console) then a possible RT=MI is reset
   c. if RT=MI (Master console) then a possible RT=MA is reset
2. The following routing codes will be set automatically by the running task or by the starting command (PSTART, POFFLOAD) when starting the task in the TCB default Routing code field TCMRTDF:
   TA (3 = Tape Pool):
   + POFFLOAD
   UR (7 = Unit Record Pool):
   + PSTART for all tasks with LST or PUN processing
   TP (8 = Teleprocessing Control):
   + PSTART for all RJE and PNET tasks
   + all RJE tasks started by IPW$$LM
   + all RJE tasks started by IPW$$SN
   + all PNET tasks started by IPW$$LD3,IPW$$LD4
   PG (11 = Programmer console):
   + PSTART partition (all IPW$$XRE and attached IPW$$XWE tasks,
   i.e. all messages from IPW$$XRE,IPW$$XWE,IPW$$XJ,IPW$$OP
3. Task termination IPW$$TR routine will set for all its messages the default routing code:
   MI (2 = Master Console Information) indicated by "(MI)*" in the TCB default routing code field TCMRTDF to insure that if the message is for an executing job (i.e. RT=PG has been set by IPW$$XRE/IPW$$XWE) which may be routed a User console (ECHO support), then a termination error message will also be routed to the Master console.
4. VSE Default EXCP/SVC0 Interface: Messages issued via EXCP/SVC0 will receive the following default handling: (except for messages in the "Exception List"):

```
Message
Number:  Routing:
-------  ----------------------
"nnnnI"  Master Console(Info  )
"nnnnD"  Master Console(Action)
"nnnnA"  Master Console(Action)
```

5. Default WTO/WTOR routing codes enforced at execution time by module IPW$$MS:

```
                   DECISION
          COMMAND    MESSAGE
 TCMRT   RESPONSE  (IPW$WTR) >>ROUTING CODE<<
+=====+===========+=========+====================+
|  0  |     NO    |   NO    |    MI              |
|     |           |         |MASTER CONSOLE INFO |
+-----+-----------+---------+--------------------+
|  0  |     NO    |   YES   |    MA              |
|     |           |         |MASTER CONSOLE ACTIO|
+-----+-----------+---------+--------------------+
|  0  |     YES   |   NO    |(ROUTING TO ORIGIN) |
+-----+-----------+---------+--------------------+
|  0  |     YES   |   YES   |(ROUTING TO ORIGIN) |
|     |           |         | + MA               |
|     |           |         |MASTER CONSOLE ACTIO|
+-----+-----------+---------+--------------------+
| NNNN|     NO    |   NO    | (ROUTING SPECIFIED)|
|     |           |         | + MI               |
|     |           |         |MASTER CONSOLE info |
+-----+-----------+---------+--------------------+
| NNNN|     NO    |   YES   | (ROUTING SPECIFIED)|
|     |           |         | + MA               |
|     |           |         |MASTER CONSOLE ACTIO|
+-----+-----------+---------+--------------------+
| NNNN|     YES   |   NO    |(ROUTING TO ORIGIN) |
|     |           |         |+(ROUTING SPECIFIED)|
+-----+-----------+---------+--------------------+
| NNNN|     YES   |   YES   |(ROUTING TO ORIGIN) |
|     |           |         |+(ROUTING SPECIFIED)|
|     |           |         | + MA               |
|     |           |         |MASTER CONSOLE ACTIO|
+-----+-----------+---------+--------------------+
|NOTE: RT=MI AND RT=MA ARE MUTUALLY EXCLUSIVE    |
+-----+-----------+---------+--------------------+

NOTE:  FOR CONNECTED COMMAND RESPONSE MESSAGES(EXCEPT
       FOR THE FIRST MESSAGE) INDIVIDUAL MESSAGE
       ROUTING CODES ARE IGNORED.
```

Descriptor Code and Color:: The developer should consider the following when deciding on how to code a message descriptor code (for an overview of the descriptor codes see Figure 56 on page 146):

Explicit Descriptor Code Needed:

- (Red:) SF (1 = System Failure)
  i.e. Hardware errors, Software logic and generation errors
- (Red:) AK (11 = Action(type b) Error Message (not DOM'd))
- (White:) DA (2 = Action(type a: DOM'd) Message)
- (Green:) II (12 = Important Information Message) e.g. TP errors

Descriptor Codes Set Automatically: The following descriptor Codes will be automatically indicated by IPW$$MS CAMS+24:)

- (Green:) CM (5 = Command Response) will be set if VSE Attention Command
- (Green:) JS (6 = Job Status)   will be set if RT=PG (Programmer Info)

Note: JS causes the message to receive the execution partition ID from VSE.

The VSE/POWER task dispatcher stores the partition ID of an execution processor in the CAT whenever it is dispatched, which is read by the supervisor console message support everytime a message is issued from VSE/POWER with the Discriptor Code = JS, causing the partition ID to be displayed with the message.

- (White:) DA (2 = decision/action) will be set if reply expected (IPW$WTR)
- (Green:) SS (4 = System Status)   will be set if nothing else is indicated
  (The White color is automaticlly set by VSE for all Reply messages WTOR)

Default WTO/WTOR descriptor codes enforced at execution time by module IPW$$MS:

```
            DECISION-                         HELD
       CMD  MSG:                               ON
 TCMDC RSP:($WTR >>DESCRIPTOR CODE<<  COLOR  TUBE
+=====+====+====+====================+=======+====+
|  0  | NO | NO |     SS             |GREEN  |NO  |
|     |    |    | SYSTEM STATUS      |       |    |
+-----+----+----+--------------------+-------+----+
|  0  | NO |YES |     DA             |WHITE  |YES |
|     |    |    | DECISION OR ACTION |HI-LITE|    |
+-----+----+----+--------------------+-------+----+
|  0  |YES | NO |     CM             |GREEN  |NO  |
|     |    |    | COMMAND RESPONSE   |       |    |
+-----+----+----+--------------------+-------+----+
|  0  |YES |YES |     DA + CM        |WHITE  |YES |
|     |    |    | DECISION OR ACTION |HI-LITE|    |
|     |    |    | (NOTE 2)           |       |    |
+-----+----+----+--------------------+-------+----+
| NN  | NO | NO |(AS SPECIFIED) NOTE1|(AS    |<-- |
|     |    |    |                    |SPEC'D)|    |
+-----+----+----+--------------------+-------+----+
| NN  | NO |YES |     DA             |WHITE  |YES |
|     |    |    | DECISION OR ACTION |HI-LITE|    |
+-----+----+----+--------------------+-------+----+
| NN  |YES | NO |(AS SPECIFIED) NOTE2|(NOTE2)|<-- |
|     |    |    |  +  CM             |       |    |
|     |    |    | COMMAND RESPONSE   |       |    |
+-----+----+----+--------------------+-------+----+
| NN  |YES |YES | DA + CM (NOTE2)    |WHITE  |YES |
|     |    |    | DECISION OR ACTION |HI-LITE|    |
|     |    |    |+COMMAND RESPONSE   |       |    |
+-----+----+----+--------------------+-------+----+
| NN  |YES |(NA)| CM + CRITICAL EVENT|RED    |YES |
|     |    |    |(ACTON TYPE B)(NOTE3)       |    |
+-----+----+----+--------------------+-------+----+
NOTE:  FOR CONNECTED COMMAND RESPONSE MESSAGES(EXCEPT
       FOR THE FIRST MESSAGE) INDIVIDUAL MESSAGE
       DESCRIPTOR CODES ARE IGNORED.
NOTE1: DC=CM MAY NOT BE SPECIFIED
              (SET AUTOMATICALLY:
               DC=CM SET WHEN COMMAND PROCESSOR OR
                         PRINT STATUS TASK IPW$$PS)
NOTE2: WHEN MULTIPLE DESCRIPTOR CODES ARE SPECIFIED
       THEN THE LOWEST CODE IS RECOGNIZED
NOTE3: WHEN COMMAND RESPONSE OCCURS WITH ACTION
       TYPE B MSG (CRITICAL EVENT) THEN COMMAND
       RESPONSE CODE IS DROPPED.
```

VSE Default EXCP/SVC0 Interface:: Messages issued via EXCP will receive the following default handling (except for messages in the "Exception List"):

Message Number:	Descriptor Code:	Colour:	Hold:
"nnnnI"	System Status (4)	green	no hold on screen
"nnnnD"	decision/action (2)	WHITE (hi-lite)	hold on screen
"nnnnA"	decision/action (2)	WHITE (hi-lite)	hold on screen

Corelation between Routing and Descriptor Codes

```
+------------------------------------------------+
|  IF ROUTING CODE = PROGRAMMER INFO (RT=PG)      |
|     DESCRIPTOR CODE SET TO JOB STATUS (DC=SS)   |
+------------------------------------------------+
|  IF DESCRIPTOR CODE = IMMED ACTION (DC=DA) THEN |
|     ROUTING CODE SET TO MASTER ACTION (RT=MA)   |
+------------------------------------------------+
```

Command Response (AR):: Any message returned from a command module IPW$$Cxx or IPW$$PS to origin console in response to an Attention Routine command, except for system errors (1QB5I,1QZ0I) which should be routed to both central operator (IPW$GAM DEST=LOCAL) and origin.

DOM'ed: Indicates if a message is deleted from the console screen via the DOM macro.

Issued Via:: For many messages it is indicated which macro is used to issue the macro, especially if it is issued via EXCP/SVC0, WTO/WTOR or IPW$WTO/IPW$WTOR. This table should be maintained accurately for all new messages.

The table uses the following abbreviations:

```
     LOC = Remote              message text locally defined in module IPW$$MS
EXCP LOC = Issued via EXCP with message text locally defined in module
EXCP REQ = Issued via EXCP with message retrieved via IPW$GAM REQ=ADDR
EXCP REQ*= Issued via EXCP with message retrieved via simulated IPW$GAM REQ=ADDR
EXCP $$I7= Issued via EXCP with message retrieved from DMB pointer constructed by IPW$$I7
WTO1 LOC = Issued via WTO  in module with message locally defined
WTO1 LOC*= Issued via WTO  in module in IPW$$I1
                              with message text locally defined in same module
WTO1 LIP*= Issued via WTO  in module in IPW$$I1
                              with message text defined in module IPW$$IP
$GAM D=L = Issued via IPW$GAM DEST=LOCAL or IPW$GAM DEST=(Rx) to local operator
$GAM+$WTO= Issued via IPW$GAM DEST=RETURN/address or IPW$GAM REQ=ADDR
                      followed by IPW$WTO to central operator
$GAM+$WTR= Issued via IPW$GAM DEST=RETURN/address/(Rx) or IPW$GAM REQ=ADDR
                      followed by IPW$WTR to central operator
$GTO     = Issued via IPW$GTO with WTO macro issued in IPW$$TS
$GTS     = Issued via IPW$GTS with WTO macro issued in IPW$$SS
$NTY     = Issued via IPW$NTY
$WTO LOC = Issued via IPW$WTO with message text locally defined in same module
$WTO LIP*= Issued via IPW$WTO with message text defined in module IPW$$IP
$WTO LCI*= Issued via IPW$WTO with message text defined in module IPW$$CI
$WTO LCD*= Issued via IPW$WTO with message text defined in module IPW$$CD
```

Different Message Types: The following table illustrates the different message types issued to the central operator. It serves as an example of how to select a message type for a new message.

```
    ========+==========+=====+=+===+=========+===============================================================+=+=====
    RT=     |DC=       |Color|C|DOM|Issued   |                                                                 |
    xx xx xx|xx xx xx  |     |d|   |         |                                                                 |
    +=======+==========+=====+=+===+=========+===============================================================+=+=====
 1. >VSE Exception Msg<|RED  | | - |EXCP --- | 1xxxI  EXCP            message                                 | |
 2. >VSE EXCP Default <|WHITE| | - |EXCP LOC | 1xxxD/1xxxA EXCP decision/action message                       | |
 3. >VSE EXCP Default <|green| | - |EXCP LOC | 1xxxI  EXCP Information message                                | |
 4.(MI)SP  *|SF       +|RED  | | - |$GAM/$WTO| Error Message                                                  | |
 5. MA SP  *|AK       +|RED  | | - |$GAM/$WTO| Action (type b) message (1QF7A)   (Note 1)                     | |
 6.(MA)    *|  (DA)   +|WHITE| | / |$GAM/$WTR| 1xxxD decision message                                         | |
 7. MA     *|   DA    +|WHITE| |Yes|$GAM/$WTO| Action (type a) message           (Note 1)                     | |
 8.(MI)SP  *|      II +|green| | - |$GAM/$WTO| Important Info message for System Programmer(e.g. 1Q0BI,1QF4I) | |
 9.(MI)SP  *|         +|green| | - |$GAM/$WTO| Important Note for System Programmer (1Q66I)                   | |
10.(MI)    *|      II +|green| | - |$GAM/$WTO| Important Info message (e.g. 1QF0I and TP errors)              | |
11.(MI)    *|         +|green| | - |$GAM/$WTO| Normal    Info message (e.g. 1Q12I)                            | |
12.(MA)    *|(DA)(CM) +|WHITE|x| - |$GAM/$WTR| Command Response message requiring Reply                       | |
13.        *|    (CM) +|green|x| - |$GAM/$WTO| Normal Command Response message                                | |
           |          |
           |          +------- "+" = additional descriptor code may be added:
           |                         -  explicitedly by IPW$GMD DC=xx
           |                         -  or by IPW$$MS default enforcement
           |
           +------------------ "*" = additonal routing may be added:
                                     - explicitedly by IPW$GMD RT=xx,
                                     - or by IPW$$MS default enforcement
                                     - or during execution by the task or PSTART command setting the default TCMRTDF

 Note 1 : Because Type A and B Action messages are not issued via IPW$WTR (ie not a decision message) then
             the routing code must be indicated explicitedly (=MA).
          Type B are RED because DC=AK specified. For Type A, DC=DA causes the WHITE colour.
```

Special Message Types (central operator only):

Note: these messages should always have a non-default routing code:

1. Short-on-resource message
   - Reason: to insure that message is routed to Master console or System Programmer Console in circumstances where a task has some default routing in effect which would cause the SOS message not to be routed to the central operator. (e.g. Execution Processor uses default routing RT=PG causing IPW$$MS to set the descriptor code DC=JS causing job execution messages to be routed to the User console by default when *$$JOB ECHO= is specified).
   - Exceptions:
     - decision/reply messages, which always get RT=MA enforced
     - Command response messages

       Reason:
       - one can assume that the person issuing the command will note the response message immediately, whereas system error messages might get lost
       - commands SHOULD fail gracefully and not cause the system to hang
   - RT=(default) which get RT=MI per default if MA enforced

```
 =========+========+=====+=+===+=========+==============================================================+=+=========+
 RT=      |DC=     |Color|C|DOM|Issued   |                    (+) Module has IPW$GMM msg EQUATE  $1xxxx ---+|         |
          |        |     |m|'ed| via:    |Message:            (o) Module has locally defined message       || Module: |
          |        |     |n|   |         |      +--- Msg equate suffix  $1xxx(n) if multiple messages     ||         |
 xx xx xx |xx xx xx|     |d|   |         |      v                                                        v|         |
 +========+========+=====+=+===+=========+==============================================================+=+=========+
 MI SP    |SF      |RED  | | - |WTO1 LOC*| 1Q03I   INSUFFICIENT REAL/PFIXED STORAGE ALLOCATED           |o|IPW$$I1  |
 MI SP    |SF      |RED  | | - |WTO1 LOC*| 1Q05I   PAGEABLE AREA nnnK TOO SMALL                         |o|IPW$$I1  |
 MI SP    |SF      |RED  | | - |WTO1 LOC*| 1Q1BI   GETVIS MACRO CALL FAILED, RC=NNNN, AREA NNK TOO SMALL |o|IPW$$I1  |
 MI SP    |SF      |RED  | | - |$GAM D=L | 1Q1DI   INSUFFICIENT GETVIS SPACE FOR QUEUE FILE, NEEDED ...  |+|IPW$$I3  |
 MI SP    |SF      |RED  | | - |WTO1 LOC*| 1Q26I   GETVIS AREA TOO SMALL                                |o|IPW$$I1  |
 MI       |      II|green| | - |         | 1Q31I   ACCOUNT FILE (IJAFILE) MORE THAN 80% FULL            |+|IPW$$AM  |
 MA       |   DA   |WHITE| |Yes|         | 1Q32A   NO MORE ACCOUNT FILE (IJAFILE) SPACE FOR task,cuu     |+|IPW$$PF  |
 MA SP    |AK      |red  | |Yes|         | 1Q38A   NO DASD SPACE AVAILABLE FOR task,cuu                 |+|IPW$$QM  |
    SP (or)|  (CM)II|green|x| - |         | 1Q7AI   commandcode NO VIRTUAL STORAGE AVAILABLE             |+|IPW$$CB  |
 MI SP    |      II|green| | - |         | 1Q78I(0)NO REAL/PFIXED STORAGE AVAILABLE FOR task,cuu        |+|IPW$$AM  |
 MI SP (or)|  (CM)II|green|X| - |         | 1Q7BI   commandcode  NO REAL/PFIXED STORAGE AVAILABLE        |+|IPW$$CD  |
>VSE EXCP Default < |green| | - |EXCP LOC | 1Q85I   task,cuu WAITING FOR VIRTUAL STORAGE                  |o|IPW$$NU  |
 MI SP    |      II|green| | - |         | 1QA6I(0)NO STORAGE AVAILABLE FOR ttttt, cuu                  |+|IPW$$AS  |
 MI SP    |      II|green| | - |         | 1QF0I   DATA FILE nnn% FULL - QUEUE FILE nnn% FULL            |+|IPW$$QM  |
 MI SP    |      II|green| | - |         | 1QF4I   NO FREE QUEUE RECORD AVAILABLE FOR task,cuu          |+|IPW$$QM  |
 MI SP    |      II|green| | - |         | 1V01I   NO SUBTASK AVAILABLE FOR RJE/SNA                  (C)|+|IPW$$SN  |
 MI SP    |      II|green| | - |         | 1V16I   NO STORAGE AVAILABLE FOR task FOR luname remid   (C)|+|IPW$$OB  |
 MI    (or)|  (CM)II|green| | - |         | 1QT9I  TRACE FACILITY TABLE LOAD IGNORED,                    |+|IPW$$TC  |
          |        |     | | - |         |          SUBTASK FAILURE                                     | |         |
                      A
                      |
                      +---- Note: to have the above messages issued with the RED color requires:
                                  - issuing DOM later to delete messages since then can occur often and should be
                                    be deleted automatically by VSE/POWER. This requires that 1Q85I  EXCP macro
                                    be replaced with WTO and DOM macros.
```

2. Severe Error Messages 1QB5I,1QZ0I
   Reason:
   To insure that a message is routed to Master console or System Programmer console in circumstances where a command response message is being issued which is normally returned to the origin console and not routed to the central operator (to route to both the Master/System console and origin console then the message must be issued twice since the descriptor codes SF + CM are mutually exclusive).

```
 =========+========+=====+=+===+=========+==============================================================+=+=========+
 RT=      |DC=     |Color|C|DOM|Issued   |                    (+) Module has IPW$GMM msg EQUATE  $1xxxx ---+|         |
          |        |     |m|'ed| via:    |Message:            (o) Module has locally defined message       || Module: |
          |        |     |n|   |         |      +--- Msg equate suffix  $1xxx(n) if multiple messages     ||         |
 xx xx xx |xx xx xx|     |d|   |         |      v                                                        v|         |
 +========+========+=====+=+===+=========+==============================================================+=+=========+
(or) SP   |SF      |RED  | | - |$GAM D=L | 1QB5I   INTERNAL MACRO CALL FAILED IN PHASE=xxxxxxxx, RC=rrmm | |         |
(or)SP -PG |SF      |RED  | | - |$GAM D=L | 1QZ0I   SEVERE LOGIC ERROR OCCURRED IN PHASE=nnnnnnnn, RC=xxxx | |         |
```

Special Message Notes

VSE "Exception List" Messages for VSE/POWER: VSE maintains a list of VSE/POWER messages (module IJBCSS00) that require special handling when issued via EXCP/SVC0 (the descriptor code is set to "system failure"):

```
                                       Issuing
        Message: Text:                 Modules: Handling:
        -------  ---------------------  -------  --------
1Q05I   PAGEABLE AREA nnnK TOO SMALL    IPW$$IN  Color: Red, Delete: Manually
1Q15I   PHASE phasename NOT FOUND       IPW$$IP     "           "
1Q2CI   PSW=xxxxxxxx, CC= (prog.check)  IPW$$AT     "           "
1Q2DI   VSE/POWER CANCELLED DUE TO ...  IPW$$AT     "           "
1QB5I   INTERNAL MACRO CALL FAILED ...  IPW$$TI     "           "
 "                                      IPW$$MS     "           "
 "                                      IPW$$I1     "           "
 "                                      IPW$$CM     "           "
1RTUI   TCP/IP INTERFACE .. TERMINATED ..  IPW$$AT     "           "
1RVUI   TCP/SSL INTERFACE QUESTIONABLE ..  IPW$$AT     "           "
```

Command Response Messages (IPW$$Cx, IPW$$PS) To support connected message function use central routine normally to issue message

- subroutine MSG in IPW$$CM
- subroutine PSMSG in IPW$$PS for display message
- subroutine PSMSGLOC in IPW$$PS for central operator message
- use IPW$GAM DEST=LOCAL for error messages 1QB5I, IQZ0I

Message Reference Table (having Message Number)

Note: Whenever the field "task" appears in a message, the same text in the code listing appears as "ttttt" which is replaced by the task-ID before printing.

Figure 136 (Page 1 of 22). Message Reference

Routing Code RT= xx xx xx	Descrip- tor Code DC= xx xx xx	Color	C m n d	DOM 'ed	Issued via:	Message: (E) Module has IPW$GMM msg EQUATE $1xxxx ----------------+ (L) Module has locally defined message +--- Msg equate suffix $1xxx(n) if multiple messages v	 | | v	Module: (where issued)
MI SP	SF	RED		-	WTO1 LOC*	1Q01I VSE/POWER CANNOT RUN IN REAL MODE	L	IPW$$I1
MI SP	SF	RED		-	WTO1 LOC*	1Q02I VSE/POWER CANNOT RUN AS A SUBTASK	L	IPW$$I1
MI SP	SF	RED		-	WTO1 LOC*	1Q03I INSUFFICIENT REAL/PFIXED STORAGE ALLOCATED	L	IPW$$I1
(MI)SP	SF	RED		-	$GAM D=L		E	IPW$$I2
(MI)SP	SF	RED		-	$GAM D=L		E	IPW$$I3
(MI)SP	SF	RED		-			E	IPW$$I4
(MI)SP	SF	RED		-			E	IPW$$I5
(MI)SP	SF	RED		-	$GAM D=L		E	IPW$$I7
(MI)SP	SF	RED		-	$GAM D=L		E	IPW$$IP
(MI)SP	SF	RED		-	$GAM D=L	1Q04I QUEUE/DATA FILE MISMATCH	E	IPW$$I3
(MI)SP	SF	RED		-			E	IPW$$I4
MI SP	SF	RED		-	WTO1 LOC*	1Q05I PAGEABLE AREA nnnK TOO SMALL	L	IPW$$I1
>VSE Exception Msg<		RED		-	EXCP LOC		L	IPW$$IN
(MI)SP	SF	RED		-	$GAM D=L	1Q06I nnn SET OR DEFINE STATEMENT(s) IGNORED	E	IPW$$I7
(MI)SP	SF	RED		-	$GAM D=L	1Q07I INVALID LOGICAL UNIT filename	E	IPW$$IP
(MI)SP	SF	RED		-			E	IPW$$I3
(MI)SP	SF	RED		-			E	IPW$$I4
(MI)SP	SF	RED		-			E	IPW$$I5
(MI)SP	SF	RED		-		1Q08I(0)UNABLE TO INITIALIZE CROSS PARTITION SUPPORT, RC=NNNN	E	IPW$$XM
(MI)SP	SF	RED		-	$GAM D=L	(1)UNABLE TO INITIALIZE NETWORKING FUNCTION, RC=NNNN	E	IPW$$IN
(MI)SP	SF	RED		-	$GAM D=L	1Q09I INVALID DEFINE STATEMENT, RC=nnnn	E	IPW$$I2
(MI)SP	SF	RED		-	$WTO LOC	1Q0AI USE PLOAD COMMAND TO LOAD (JOBEXIT|NETEXIT ...	L	IPW$$IN
MI SP	SF	RED		-	WTO1 LOC*		L	IPW$$I1
(MI)SP	II	green		-		1Q0BI DATA FILE TOO LARGE	E	IPW$$I4
(MI)SP	SF	RED		-	$GAM D=L	1Q0CI IJQFILE TOO LARGE, nnnnn QUEUE RECORDS UNUSED	E	IPW$$I3

Figure 136 (Page 2 of 22). Message Reference

Routing Code RT= xx xx xx	Descrip- tor Code DC= xx xx xx	Color	C m n d	DOM 'ed	Issued via:	Message: (E) Module has IPW$GMM msg EQUATE $1xxxx (L) Module has locally defined message +--- Msg equate suffix $1xxx(n) if multiple messages v	 v	Module: (where issued)
(MI)SP	SF	RED		-		1Q0DI ACCOUNT FILE TOO SMALL, REQUIRED BLOCKS=nnn	E	IPW$$I5
(MI)SP	SF	RED		-	$GAM D=L	1Q0EI ACCOUNT SUPPORT NOT AVAILABLE	E	IPW$$I2
(MI)SP	SF	RED		-		1Q0FI DATA FILE SPECIFICATION ERROR, RC=nnnn	E	IPW$$I4
MA	DA	WHITE		-	WTO1 LOC*	1Q0GA Current LEVEL v.rm OF VSE/POWER INCOMPATIBLE ...	L	IPW$$I1
(MA)SP	SF	RED		-	$GAM+$WTR	1Q0HD IF SPOOL FILE MIGRATION TO	E	IPW$$I3
(MI)		GREEN		-	$GAM D=L	1Q0HI CURRENT LEVEL ..... OF VSE/POWER	E	IPW$$I3
MA	DA	WHITE		-	$GAM D=L	1Q0JA SPOOL FILE MIGRATION FAILED DUE TO	E	IPW$$I3
(MI)		GREEN		-	$GAM D=L	1Q0KI(0)DATA FILE EXTENT NO. ... AS EXTRACTED	E	IPW$$I4
(MI)		GREEN		-	$GAM D=L	(1)DATA FILE EXTENT NO. ... AS PRESERVED	E	IPW$$I4
MI SP	SF	RED		-	WTO1 LOC*	1Q10I SUPERVISOR WITHOUT ACCOUNTING SUPPORT	L	IPW$$I1
(MA)	(DA)	WHITE		-	$GAM+$WTR	1Q11D FORMAT QUEUES=	E	IPW$$I2
(MI)		green		-	$GAM+$WTO	1Q12I VSE/POWER 7.1.0 INITIATION COMPLETED (FOR SYSID n)	E	IPW$$I7
(MI)SP	SF	RED		-	$GAM D=L	1Q13I ERRONEOUS AUTOSTART CARD(S) READ	E	IPW$$I2
MI SP	SF	RED		-	WTO1 LOC*	1Q14I NO MATCHING PUB FOR cuu	L	IPW$$I1
(MI)SP	SF	RED		-	$WTO LIP*	1Q15I PHASE phasename NOT FOUND	L	IPW$$IN
MI SP	SF	RED		-	WTO1 LIP*	(Message locally defined in IPW$$IP)	L	IPW$$I1
>VSE Exception Msg<		RED		-	EXCP LOC		L	IPW$$IP
(or)	(CM)	green	x	-	$GAM+$WTO	1Q15I(1)commandcode PHASE phasename NOT FOUND	E	IPW$$CLD
(or)	(CM)	green	x	-	$WTO LOC	1Q15I UNABLE TO LOAD xxxxxxxx xxxxxxxx RC=	L	IPW$$CLD
(MI)SP	SF	RED		-	$WTO LIP*	(Message locally defined in IPW$$IP and IPW$$CLD)	L	IPW$$IN
MI SP	SF	RED		-	WTO1 LIP*		L	IPW$$I1
MI SP	SF	RED		-	WTO1 LOC*	1Q16I INVALID PUN|LST ROUTING FOR REMID	L	IPW$$I1
(MI)SP	SF	RED		-	$GAM D=L	1Q17I QUEUE FILE TOO SMALL	E	IPW$$I3
(MI)SP	SF	RED		-		1Q18I TOO MANY DATA FILE EXTENTS	E	IPW$$I4
(MI)SP	SF	RED		-		1Q19I INVALID DATA FILE EXTENT, RC=NNNN	E	IPW$$I4
(MI)		green		-	$GAM D=L	1Q1AI INVALID DEVICE SPECIFICATION CUU, RC=NNNN	E	IPW$$OT
MI SP	SF	RED		-	WTO1 LOC*	1Q1BI GETVIS MACRO CALL FAILED, RC=NNNN, AREA NNK TOO SMALL	L	IPW$$I1
(MI)SP	SF	RED		-		1Q1CI DBLK SIZE MISMATCH: DATA FILE=XXXX, POWER MACRO=NNNN	E	IPW$$I4
MI SP	SF	RED		-	$GAM D=L	1Q1DI INSUFFICIENT GETVIS SPACE FOR QUEUE FILE, NEEDED ...	E	IPW$$I3
(MI)SP	II	green		-	$GAM D=L	1Q1EI ATTEMPTING TO PLACE QUEUE FILE INTO PARTITION GETVIS .	E	IPW$$I3
(MI)SP	SF	RED		-		1Q1FI DBLK GROUP MISMATCH: DATA FILE=..., POWER MACRO=...	E	IPW$$I4
(MI)		green		-	$GAM D=L	1Q20I AUTOSTART IN PROGRESS	E	IPW$$I7
(MI)SP	II	green		-	$GAM D=L	1Q21I VSE/POWER HAS BEEN TERMINATED	E	IPW$$T1
MI SP	SF	RED		-	WTO1 LOC*	1Q22I VSE/POWER ALREADY ACTIVE	L	IPW$$I1
(MI)SP	SF	RED		-	$GAM D=L	1Q23I LTA CANCEL IN PHASE=xxxxxxxx	E	IPW$$I2
(MI)SP	SF	RED		-	$GAM D=L		E	IPW$$I3
(MI)SP	SF	RED		-			E	IPW$$I4
(MI)SP	SF	RED		-			E	IPW$$I5
(MI)SP	II	green		-	$GAM D=L	1Q24I ATTEMPTING TO PLACE QUEUE FILE INTO VIO AREA	E	IPW$$I3
MA	DA	WHITE		Yes	$GAM+$WTO	1Q25A partition-id IN STOP STATE	E	IPW$$T1
(VSE EXCP Default <		WHITE		-	EXCP LOC	1Q25D SUGGEST TO TAKE STAND ALONE DUMP NOW ...	L	IPW$$AT

Figure 136 (Page 3 of 22). Message Reference

Routing Code RT= xx xx xx	Descrip- tor Code DC= xx xx xx	Color	C m n d	DOM 'ed	Issued via:	Message: (E) Module has IPW$GMM msg EQUATE $1xxxx ----------------+ (L) Module has locally defined message +--- Msg equate suffix $1xxx(n) if multiple messages v	 v	Module: (where issued)
>VSE EXCP Default <		green		-	EXCP LOC	1Q25I CLEANUP PENDING FOR	E	IPW$$AT
>VSE EXCP Default <		green		-	EXCP LOC	HANDLE OUTSTANDING REQUESTS	E	IPW$$AT
>VSE EXCP Default <		green		-	EXCP LOC	RECURSIVE ENTRY OF TERMINATION	E	IPW$$AT
MI SP	SF	RED		-	WTO1 LOC*	1Q26I GETVIS AREA TOO SMALL	L	IPW$$I1
(MI)SP	SF	RED		-	$GAM D=L		E	IPW$$I2
(MI)SP	SF	RED		-			E	IPW$$I4
(MI)SP	SF	RED		-			E	IPW$$I5
(MI)SP	SF	RED		-	$GAM D=L		E	IPW$$I7
(MI)SP	SF	RED		-	$GAM D=L	1Q27I UNABLE TO INITIALIZE SPOOL MANAGEMENT	E	IPW$$I7
(MI)		green		-	$GAM D=L	1Q28I END OF VOLUME ON cuu	E	IPW$$OT
(MI)		green		-		1Q29I END OF INPUT ON cuu	E	IPW$$SY
(MI)		green		-	$GAM D=L	1Q2AI OFFLOADING SUCCESSFULLY COMPLETED ON cuu	E	IPW$$OF
(MI)		green		-	$GAM D=L	1Q2BI(0)NOTHING TO SAVE ON cuu, RC=	E	IPW$$OF
(MI)		green		-	$GAM D=L	(1)NOTHING TO SELECT ON cuu, RC=	E	IPW$$OF
>VSE Exception Msg<		RED		-	EXCP LOC	1Q2CI PSW=XXXXXXXXXXXXXXXX, CC=yy -progr. check desc.	L	IPW$$AT
>VSE Exception Msg<		RED		-	EXCP LOC	1Q2DI VSE/POWER CANCELED DUE TO PROGRAM REQUEST IN PHASE=xxx	L	IPW$$AT
>VSE EXCP Default <		WHITE		-	EXCP LOC	1Q2ED SPECIFY PRINTER OR TAPE FOR VIO STORAGE COPY OF	L	IPW$$AT
>VSE EXCP Default <		WHITE		-	EXCP LOC	INVALID PRINTER/TAPE, RE-ENTER	L	IPW$$AT
>VSE EXCP Default <		green		-	EXCP LOC	1Q2FI VIO POINT PROCESSING FAILED RC=nn	L	IPW$$AT
>VSE EXCP Default <		green		-	EXCP LOC	1Q2GI NORMAL TERMINATION OF QUEUE FILE DUMP	L	IPW$$AT
(or)	(CM)	green	x	-	$GAM+$WTO	(0)NORMAL TERMINATION OF QUEUE FILE DUMP	E	IPW$$PS
(or)	(CM)	green	x	-	$GAM+$WTO	(1)QUEUE FILE DUMP PROCESSING CANCELED BY OPERATOR	E	IPW$$PS
(or)	(CM)	green	x	-	$GAM+$WTO	(2)NORMAL TERMINATION OF QUEUE DUMP, SEE LIST ENTRY $VIO	E	IPW$$PS
(or)	(CM)	green	x	-	$GAM+$WTO	(3)NORMAL TERMINATION OF QUEUE DUMP, SEE LIST ENTRY $QFL	E	IPW$$PS
>VSE EXCP Default <		green		-	EXCP LOC	1Q2HI <JOB|NET|OUT|XMTEXIT>=phasename PUT INTO FAILED STATE	L	IPW$$AT
>VSE EXCP Default <		green		-	EXCP LOC	1Q2JI IDUMP REQUEST FROM X'ADDRESS' BY ...	L	IPW$$ID
>VSE EXCP Default <		green		-	EXCP LOC	1Q2KI VSE/POWER RECOVERING FROM FAILURE OF USER EXIT	L	IPW$$AT
(MI) TA		green		-	$GAM D=L	1Q2LI POFFLOAD ON cuu HAS DETECTED AN INCORRECT SPOOL ENTRY..	E	IPW$$OF
(or)	(CM)	green	x	-	$WTO LOC	1Q2MI PDISPLAY BIGGEST DETECTED QUEUE RECORD WITH INVALID...	L	IPW$$PS
>VSE EXCP Default <		WHITE		-	EXCP LOC	1Q30D ABNORMAL VSE/POWER TERMINATION, DUMP REQUIRED ?(YES/NO)	L	IPW$$AT
MI	II	green		-		1Q31I ACCOUNT FILE (IJAFILE) MORE THAN 80% FULL	E	IPW$$AM
(MI)	II	green		-			E	IPW$$PF
MA SP	DA	WHITE		Yes		1Q32A NO MORE ACCOUNT FILE (IJAFILE) SPACE FOR task,cuu	E	IPW$$PF
MA SP	DA	WHITE		Yes			E	IPW$$AM
(MI)		green		-		1Q33I(0)STOPPED partition-id	E	IPW$$XRE
(MI)*		green		-	$GAM+$WTO	(1)STOPPED task,cuu	E	IPW$$TR
MA	DA	WHITE		Yes		1Q34A partition-id WAITING FOR INPUT ON cuu	E	IPW$$XRE
(MI)		green		-		1Q34I(0)partition-id WAITING FOR WORK	E	IPW$$XRE
(MI)		green		-		(1)task WAITING FOR WORK ON cuu	E	IPW$$LW
(MI)		green		-			E	IPW$$PR
MA	DA	WHITE		Yes		1Q35A UNEXPECTED EOF ON cuu	E	IPW$$PR
(MI)SP	II			-		1Q36I DISP=X JOB(S) IN VSE/POWER READER QUEUE AFTER ABEND	E	IPW$$XRE
(MI)		green		-	$GAM D=L	1Q37I JECL STATEMENT INCORRECT NEAR COLUMN n	E	IPW$$LR
(MI)		green		-	$GAM D=L		E	IPW$$NR2
MA SP	AK	red		-		1Q38A NO DASD SPACE AVAILABLE FOR task,cuu	E	IPW$$QM
(MI)		green		-		1Q39I queue jobname FLUSHED BY THE OPERATOR, VSE/POWER OR USER	E	IPW$$LO
(MI)		green		-	$GAM D=L		E	IPW$$LR

Figure 136 (Page 4 of 22). Message Reference

Routing Code RT=	Descrip-tor Code DC=	Color	Cmnd	DOM 'ed	Issued via:	Message:		Module: (where issued)
						(E) Module has IPW$GMM msg EQUATE $1xxxx ----------------+		
						(L) Module has locally defined message		
						+--- Msg equate suffix $1xxx(n) if multiple messages		
xx xx xx	xx xx xx					v	v	
(MI)		green		-			E	IPW$$LW
(MI)SP	SF	RED		-		1Q3AI ERROR WHILE PROCESSING ACCOUNT RECORD, RC=xxx	E	IPW$$PF
(MI)SP	SF	RED		-		1Q3BI DBLK SIZE SET TO TRACK CAPACITY OF	E	IPW$$I4
(MI)SP	SF	RED		-		1Q3CI INVALID BLOCKSIZE FOR filename	E	IPW$$SF
(MA)SP	SF	RED		-		1Q3DI INVALID CI-SIZE FOR filename	E	IPW$$SF
(MI)		green		-		1Q3EI DYNAMIC CLASS class WAITING FOR WORK	E	IPW$$XRE
(MI)SP	II	green		-	$GAM D=L	1Q3FI(0)DYNAMIC CLASS(ES) SUSPENDED - NO ALLOCATION SPACE	E	IPW$$DP
(MI)SP	II	green		-	$GAM D=L	(1)DYNAMIC CLASS(ES) SUSPENDED - NO MORE PARTITIONS AT ALL	E	IPW$$DP
(MI)SP	II	green		-	$GAM D=L	(2)DYNAMIC CLASS(ES) SUSPENDED - NO SYSTEM GETVIS SPACE	E	IPW$$DP
(MI)SP	II	green		-	$GAM D=L	(3)DYNAMIC CLASS(ES) SUSPENDED - NO VSE/POWER SETPFIX SP	E	IPW$$DP
(MI)SP	II	green		-	$GAM D=L	(4)DYNAMIC CLASS(ES) SUSPENDED - NO VSE/POWER GETVIS SP-24	E	IPW$$DP
(MI)SP	II	green		-	$GAM D=L	(5)DYNAMIC CLASS(ES) SUSPENDED - NO PFIXED SYSTEM GETVIS	E	IPW$$DP
(MI)SP	II	green		-	$WTO LOC	DYNAMIC CLASS(ES) SUSPENDED - ALLOC SPACE RUUING OUT	L	IPW$$DP
(MI)SP	II	green		-	$WTO LOC	DYNAMIC CLASS(ES) SUSPENDED - POWER GETVIS-24 RUNNG OUT	L	IPW$$DP
(MI)SP	II	green		-	$WTO LOC	1Q3GI RESTRICTED ALLOCATION OF DYN.PART.cn - NO SYSTEM ...	L	IPW$$DP
MA SP	AK	RED		Yes	$GAM D=L	1Q3JA NEW SAS=cccc TASK REJECTED DUE TO .. UBCHAIN	E	IPW$$XM
MA	DA	WHITE		Yes		1Q40A(0)ON taskid FORMS ffff NEEDED FOR jobname jobnumber (R)	E	IPW$$LW
MA	DA	WHITE		Yes		(1)ON cuu FORMS ffff NEEDED FOR jobname jobnumber	E	IPW$$LW
(MI)		green		-		1Q41I MISMATCHING PRINTER|PUNCH TYPE FOR jobnm jobno ON ..	E	IPW$$LW
(MI)		green		-		1Q42I(0)PAGE/CARD COUNT EXCEEDS END OF QUEUE ENTRY FOR cuu	E	IPW$$LW
(MI)		green		-		(1)PAGE/CARD COUNT EXCEEDS END OF QUEUE ENTRY FOR taskid	E	IPW$$LW
(MI) TA		green		-	$GAM D=L	1Q43I END-OF-FILE ON TAPE FOR task,cuu	E	IPW$$QM
(MI)		green		-	$GAM D=L	1Q44I INVALID OR INCOMPLETE PARAMETER COMBINATION taskid	E	IPW$$XJ
(MI)		green		-	$GAM D=L	1Q45I SLI STATEMENT NOT SUPPORTED partition-id	E	IPW$$XJ
(MI)		green		-	$GAM D=L	1Q46I DISP FORCED TO D FOR jobname jobnumber	E	IPW$$XJ
(MI)		green		-	$GAM D=L	1Q47I partition-id jobname jobnumber FROM node/user TIME=..	E	IPW$$XJ
(MI)		green		-			E	IPW$$XRE
(MI)		green		-	$GAM D=L	1Q48I NO MATCHING SPOOL DEVICE partition-id	E	IPW$$XJ
(MI)		green		-	$GAM D=L	1Q49I INVALID DELIMITER partition-id	E	IPW$$XJ
(MI)SP	II	green		-		1Q4AI(0)MESSAGE DISCARDED, RC=nnnn	E	IPW$$NS
(MI)SP	II	green		-		1Q4AI(1)MESSAGE DISCARDED, RC=nnnn APPLID USESRID	E	IPW$$NS
(MI)SP	SF	RED		-		1Q4BI NOTIFY SUPPORT CANCELED FOR nnnnnnnn	E	IPW$$NS
(MI)SP	SF	RED		-	$GAM D=L	1Q4CI UNABLE TO START VSE/POWER - NOT RUNNING IN SHARED ADDR	E	IPW$$I2
(MI)		green		-		1Q4DI JOB jobname jobnumber FINISHED PROCESSING IN PART.....	E	IPW$$XRE
(MI) SE		green		-		1Q4EI JOB jobname jobnumber pid NOT AUTH'ZED TO EXECUTE RC=.	E	IPW$$XRE
(MI) SE		green		-	$GAM D=L		E	IPW$$XJ
(MI)		green		-	$GAM D=L	1Q4FI JOB jobname jobno pid FLUSHED BY '* $$ FLS' STATEMENT	E	IPW$$XJ
(MI)		green		-		1Q4GI cuu OUTPUT NOT PURGED FOR jobname jobnumb IN PARTITION	E	IPW$$XWE
(MI) SE		green		-		1Q4HI JOB jobname jobnumber pid RUNNING IN WRONG SEC ZONE,..	E	IPW$$XRE
(MI) SE		green		-	$GAM D=L		E	IPW$$XJ
(MI) SE		green		-		1Q4JI JOB jobname jobnumber pid SECURITY USERID NOT AUTH'ZED.	E	IPW$$XRE
(MI) SE		green		-	$GAM D=L		E	IPW$$XJ
(MI)		green		-	$GAM D=L	1Q4KI nnnnn RECORDS IGNORED FOR jobname ...(msg about remote)	E	IPW$$LM
(MI)		green		-	$GAM D=L	1Q4LI nnnnn RECORDS IGNORED FOR jobname ...(msg about local)	E	IPW$$LM

Figure 136 (Page 5 of 22). Message Reference

Routing Code RT= xx xx xx	Descrip- tor Code DC= xx xx xx	Color	C m n d	DOM 'ed	Issued via:	Message: (E) Module has IPW$GMM msg EQUATE $1xxxx ----------------+ (L) Module has locally defined message +--- Msg equate suffix $1xxx(n) if multiple messages v		Module: (where issued)
(MI)		green		-	$GAM D=L	1Q50I UNKNOWN KEYWORD partition-id	E	IPW$$XJ
(MI)		green		-			E	IPW$$OP
(MI)		green		-	$GAM D=L	1Q51I INVALID paraname PARAMETER partition-id	E	IPW$$XJ
(MI)		green		-			E	IPW$$OP
(MI)		green		-		1Q52I OUTPUT LIMIT EXCEEDED FOR jobname jobnumber part-id,cuu	E	IPW$$XWE
(MI)		green		-		1Q53I OUTPUT SEGMENTED FOR jobname jobnumber part-id,cuu	E	IPW$$XWE
(MA)		WHITE		-		1Q54A FCB ERROR FOR jobname jobnumber task,cuu PHASE=.......	E	IPW$$AS
(MA)		WHITE		-			E	IPW$$XWE
(MI)		green		-	$GAM D=L	1Q54I UCS ERROR FOR jobname jobnumber task,cuu PHASE=.......	E	IPW$$PL
(MA) TA	(DA)	WHITE		-	$GAM+WTR	1Q55A(0)SPECIFY TAPE ADDRESS FOR jobname jobnumber part-id,cuu	E	IPW$$OT
(MA) TA	(DA)	WHITE		-	$GAM+WTR	(1)SPECIFY TAPE ADDRESS FOR task,cuu	E	IPW$$OT
(MI) TA		green		-	$GAM D=L	1Q56I INVALID TAPE ADDRESS/MODE SET task	E	IPW$$OT
MA TA	DA	WHITE		Yes	$GAM D=L	1Q57A(0)PLEASE REMOVE WRITE PROTECTION ON dev FOR task,cuu	E	IPW$$OT
MA TA	DA	WHITE		Yes	$GAM D=L	(1)PLEASE REMOVE WRITE PROTECTION ON dev FOR task,cuu	E	IPW$$OT
MA TA	DA	WHITE		Yes	$GAM+$WTO	1Q58A(0)MOUNT TAPE ON dev FOR jobname jobnumber task,cuu	E	IPW$$OT
MA TA	DA	WHITE		Yes	$GAM+$WTO	(1)MOUNT TAPE ON dev FOR task	E	IPW$$OT
(MI)SP	II	green		-		1Q59I task,cuu WAITING FOR REAL/PFIXED STORAGE	E	IPW$$NU
(MI) TA		WHITE		-	$GAM D=L	1Q5AI INVALID TAPE MOUNTED ON dev FOR task,cuu RC=nnnn	E	IPW$$OT
(MI) TA		WHITE		-			E	IPW$$SY
(MI) TA		green		-	$GAM D=L	1Q5BI NO TRAILER LABEL FOUND ON dev FOR task	E	IPW$$OT
(or) (TA)	(CM)	green	x	-		1Q5CI(0)commandcode MODE VERIFICATION FAILED,CURRENT MODE TAKEN	E	IPW$$CO
(or)	(CM)	green	x	-			E	IPW$$CJ
(MI) TA		green		-	$GAM D=L	(1)MODE VERIFICATION FAILED, CURRENT MODE TAKEN FOR ...	E	IPW$$OT
(MI)		green		-		1Q5DI(0)EXECUTION COMPLETED FOR jobname jobnumber ON nodeid ..	E	IPW$$XRE
(MI)		green		-			E	IPW$$NS
(MI)		green		-		(1)EXECUTION COMPLETED FOR jobname jobnumber, RC=nnnn, ..	E	IPW$$NS
(MI)		green		-			E	IPW$$XRE
(or)	(CM)	green	x	-	$GAM D=L	1Q5EI DTR$DYNx.Z INTERNAL PLOAD DYNC FAILURE, RC/FB=....	E	IPW$$CLD
(MI)		green		-	$NTY+$WTO	1Q5FI FORMATTED COMMAND PROCESSING NOT SUPPORTED	E	IPW$$MX
(MI)		green		-	$GAM D=L	1Q5GI INVALID STATEMENT FROM IPWSEGM MACRO, COL=...RC=nnnn	E	IPW$$XJ
				-	$NTY	1Q5HI >.... fixed format job generation message ... < < (never displayed on console) . <		IPW$XWE
(MI)		green		-	$WTO LOC	1Q5JI / ...... 50 bytes invalid JECL .... /	L	IPW$$XJ
(MI)SP TA	II	green		-	$GAM D=L	1Q5KI(0)TAPE SPOOLING FORCED TO DISK DUE TO BLOCKED LTA ...	E	IPW$$XWE
(MI)		green		-	$GAM D=L	1Q5KI(1)TAPE SPOOLING FORCED TO DISK ...	E	IPW$$OT
MI	II	green		-	$GAM D=L	1Q5LI(0)VSE/POWER OFFLOAD TERMINATED FOR --	E	IPW$$OF
MI	II	green		-	$GAM D=L	1Q5LI(1)VSE/POWER OFFLOAD TERMINATED FOR --	E	IPW$$TR
MI	SF	RED		-	$GAM D=L	1Q5MI(0)POFFLOAD cccccccc JOURNALING ON ...	E	IPW$$OF
							E	IPW$$TR
MI	SF	RED		-	$GAM D=L	1Q5MI(0)POFFLOAD cccccccc JOURNALING ON ...	E	IPW$$CO
MI	SF	RED		-	$GAM D=L	1Q5NI OFFLOADING ERRON ON cuu,task, JOURNAL ...	E	IPW$$TR
(MI)TA		green		-	$GAM D=L	1Q5OI(0)CARTRIDGE ON cuu ALREADY WRITTEN ONCE ...	E	IPW$$OT
(MI)TA		green		-	$GAM D=L	1Q5OI(0)CARTRIDGE ON cuu ALREADY WRITTEN ONCE ...	E	IPW$$OT
(MI)SP	II	green		-		1Q60I OPEN FAILURE ON PACCOUNT OUTPUT DEVICE	E	IPW$$AM
(MI)SP	II	green		-			E	IPW$$SF
(MA)SP	(DA)	RED		-	$GAM+$WTR	1Q61A UNRECOVERABLE I/O ERROR ON dev-des (READ|WRITE) I/O ERR	E	IPW$$PL
(MA)SP	(DA)	RED		-	$GAM+$WTR		E	IPW$$PP

Figure 136 (Page 6 of 22). Message Reference

```
 Routing  |Descrip-|Color|C|DOM|Issued   |Message:                                                          | Module: |
 Code     |tor Code|     |m|'ed| via:    |       (E) Module has IPW$GMM msg EQUATE  $1xxxx ----------------+|(where   |
 RT=      |DC=     |     |n|   |         |       (L) Module has locally defined message                    || issued) |
          |        |     |d|   |         |       +--- Msg equate suffix  $1xxx(n) if multiple messages     ||         |
 xx xx xx |xx xx xx|     | |   |         |       v                                                        v|         |

 ---------+--------+-----+-+---+---------+-----------------------------------------------------------------+-+---------+
(MI)*SP   |SF      |RED  | | - |$GAM+$WTO| 1Q61I(0)READ I/O ERROR ON dev, seek address or block number... |E|IPW$$TR  |
(MI)*SP   |SF      |RED  | | - |$GAM D=L |                                                                 |E|IPW$$RY  |
(MI)*SP   |SF      |RED  | | - |$GAM+$WTO|      (1)UNRECOVERABLE I/O ERROR ON ACCOUNT FILE cuu             |E|IPW$$TR  |
(MI)*SP   |SF      |RED  | | - |$GAM+$WTO|      (2)UNRECOVERABLE I/O ERROR ON PACCOUNT OUPUT DEVICE        |E|IPW$$TR  |
(MI)*SP   |SF      |RED  | | - |$GAM+$WTO|      (3)UNRECOVERABLE I/O ERROR ON tape                         |E|IPW$$TR  |
(MI)*SP   |SF      |RED  | | - |$GAM+$WTO|      (4)UNRECOVERABLE I/O ERROR ON cuu                          |E|IPW$$TR  |
(MI)*SP   |SF      |RED  | | - |$GAM+$WTO|      (5)WRITE I/O ERROR ON dev, seek address or block number..  |E|IPW$$TR  |
 ---------+--------+-----+-+---+---------+-----------------------------------------------------------------+-+---------+
(MI)*SP   |SF      |RED  | | - |$GAM+$WTO| 1Q62I   QUEUE CONTROL AREA UNACCESSABLE, RC=...                 |E|IPW$$TR  |
 ---------+--------+-----+-+---+---------+-----------------------------------------------------------------+-+---------+
(MI)*SP   |SF      |RED  | | - |$GAM+$WTO| 1Q63I   PERM I/O ERROR WRITING/READING QUEUE FILE MASTER RECORD|E|IPW$$TR  |
 ---------+--------+-----+-+---+---------+-----------------------------------------------------------------+-+---------+
(MI)*SP   |SF      |RED  | | - |$GAM+$WTO| 1Q64I(0)JOB jobname jobnumber queue ENTRY DELETED - nnnn DBLK.  |E|IPW$$TR  |
(MI)*SP   |SF      |RED  | | - |$GAM+$WTO|      (1)JOB jobname jobnumber queue ENTRY DELETED               |E|IPW$$TR  |
(MI)SP    |SF      |RED  | | - |$GAM D=L |                                                                 |E|IPW$$OF  |
 ---------+--------+-----+-+---+---------+-----------------------------------------------------------------+-+---------+
(MI)*SP   |SF      |RED  | | - |$GAM+$WTO| 1Q65I   JOB jobnam jobnu suf q ERRONEOUS, OPERATOR SHOULD DELET|E|IPW$$TR  |
 ---------+--------+-----+-+---+---------+-----------------------------------------------------------------+-+---------+
(MI)*SP   |        |green| | - |$GAM D=L | 1Q66I   ACCOUNT FILE KEPT                                       |E|IPW$$TR  |
 ---------+--------+-----+-+---+---------+-----------------------------------------------------------------+-+---------+
(or)      |  (CM)  |green|x| - |         | 1Q67I   NO EXIT ROUTINE CURRENTLY LOADED                        |E|IPW$$CV  |
(or)      |  (CM)  |green|x| - |         |                                                                 |E|IPW$$PS  |
 ---------+--------+-----+-+---+---------+-----------------------------------------------------------------+-+---------+
(MI)*     |        |green| | - |$GAM+$WTO| 1Q68I   SEGMENTATION FORCED FOR jobname jobnumber part-id,cuu   |E|IPW$$TR  |
 ---------+--------+-----+-+---+---------+-----------------------------------------------------------------+-+---------+
(MI)*     |        |green| | - |$GAM+$WTO| 1Q69I   DEFAULT OPTIONS TAKEN FOR jobname jobnumber part-id,cuu +|IPW$$TR  |
 ---------+--------+-----+-+---+---------+-----------------------------------------------------------------+-+---------+
(or)      |  (CM)  |green|x| - |$GAM+$WTO| 1Q6AI(0)** DISPLAY OF ACTIVE DYNAMIC CLASS TABLE DTR$DYNx.Z **  |E|IPW$$PS  |
(or)      |  (CM)  |green|x| - |$GAM+$WTO|      (1)** DISPLAY OF VERIFIED DYNAMIC CLASS TABLE DTR4DYNx.Z**|E|IPW$$PS  |
(or)      |  (CM)  |green|x| - |$GAM+$WTO|      (2)CLS STATE   ACT/MAX ALLOC    SIZE  SP-GETV PROFILE LUBS|E|IPW$$PS  |
(or)      |  (CM)  |green|x| - |$GAM+$WTO|      (3)NO MATCHING DYNAMIC CLASS FOUND                         |E|IPW$$PS  |
(or)      |  (CM)  |green|x| - |$WTO LOC |         (------------ display line   PDISPLAY DYNC ------)    |L|IPW$$PS  |
 ---------+--------+-----+-+---+---------+-----------------------------------------------------------------+-+---------+
(or)      |  (CM)  |green|x| - |$GAM+$WTO| 1Q6BI(0)DYNAMIC CLASS TABLE LOADED SUCCESSFULLY                 |E|IPW$$PS  |
(or)      |  (CM)  |green|x| - |$GAM+$WTO|      (1)DYNAMIC CLASS TABLE LOADED WITH INVALID CLASSES         |E|IPW$$PS  |
(or)      |  (CM)  |green|x| - |$GAM+$WTO|      (2)DYNAMIC CLASS TABLE NOT LOADED                          |E|IPW$$PS  |
(or)      |  (CM)  |green|x| - |$GAM+$WTO|      (3)DYNAMIC CLASS TABLE VERIFIED                            |E|IPW$$PS  |
(or)      |  (CM)  |green|x| - |$GAM D=L |      (4)DYNAMIC CLASS TABLE DISPLAYED IN LIST ENTRY $DYD....    |E|IPW$$PS  |
(or)      |  (CM)  |green|x| - |$GAM+$WTO|      (5)DYNAMIC CLASS TABLE NOT LOADED - ACTIVE CLAS MISSING    |E|IPW$$PS  |
(or)      |  (CM)  |green|x| - |$GAM+$WTO|      (6)DYNAMIC CLASS TABLE NOT LOADED - NO DYN PARTITION DEFN  |E|IPW$$PS  |
 ---------+--------+-----+-+---+---------+-----------------------------------------------------------------+-+---------+
(or)      |  (CM)  |green|x| - |         | 1Q6CI   commandcode NO ACTIVE DYNAMIC CLASS TABLE LOADED        |E|IPW$$CD  |
(or)      |  (CM)  |green|x| - |         |                                                                 |E|IPW$$CV  |
 ---------+--------+-----+-+---+---------+-----------------------------------------------------------------+-+---------+
 MA SP    |AK      |RED  | | - |$GAM D=L | 1Q6DA   RESERVED GETVIS SUBPOOL-ID IJBPxx ALREADY USED          |E|IPW$$DP  |
 ---------+--------+-----+-+---+---------+-----------------------------------------------------------------+-+---------+
(or)      |  (CM)  |green|x| - |         | 1Q6EI   CLASS class NOT DEFINED IN ACTIVE DYNAMIC CLASS TABLE   |E|IPW$$CV  |
 ---------+--------+-----+-+---+---------+-----------------------------------------------------------------+-+---------+
 MA SP    |AK      |RED  | | - |$GAM D=L | 1Q6FA   BRING UP OF DYNAMIC PARTITION partid HAS FAILED, RC=xx  |E|IPW$$DP  |
 MA SP    |AK      |RED  | | - |         |                                                                 |E|IPW$$XRE |
 ---------+--------+-----+-+---+---------+-----------------------------------------------------------------+-+---------+
 MA SP(MI)*|AK     |RED  | | - |$WTO LOC | 1Q6GA   FAILING R/W-I/O REQUEST FOR UNDEFINED DBLK=...          |L|IPW$$TR  |
 ---------+--------+-----+-+---+---------+-----------------------------------------------------------------+-+---------+
 MA SP(MI)*|AK     |RED  | | - |$WTO LOC | 1Q6HA   FAILING R/W-I/O REQUEST FOR NON-SER DBLK=...            |L|IPW$$TR  |
 ---------+--------+-----+-+---+---------+-----------------------------------------------------------------+-+---------+
 MI SP(MI)*|AK     |green| | - |$WTO LOC | 1Q6JI   JOB jobname jobnumb qid   ENTRY KEPT WITH HOLD DISP ..  |L|IPW$$TR  |
(MI)SP TA |        |green| | - |$GAM D=L |                                                                 |E|IPW$$OF  |
 ---------+--------+-----+-+---+---------+-----------------------------------------------------------------+-+---------+
 MA SP(MI)*|AK     |RED  | | - |$WTO LOC | 1Q6KA   FAILING R/W-I/O REQUEST: NO SER IN SER DBLK=...         |L|IPW$$TR  |
 ---------+--------+-----+-+---+---------+-----------------------------------------------------------------+-+---------+
 MA SP    |AK      |RED  | | - |$GAM D=L | 1Q6LA   INVALID LOGICAL RECORD LENGTH FOUND IN DBLK, TASK TERM  |E|IPW$$DM  |
 ---------+--------+-----+-+---+---------+-----------------------------------------------------------------+-+---------+
(MI)SP    |SF      |RED  | | - |$WTO LOC | 1Q6MI   taskid, cuu   INVALID LOGICAL RECORD LENGTH ...         |L|IPW$$OF  |
 ---------+--------+-----+-+---+---------+-----------------------------------------------------------------+-+---------+
(MI)   TA |        |green| | - |$GAM D=L | 1Q6NI   POFFLOAD PICKUP HAS SCHEDULED nnnn SPOOL ENTRIES ..     |E|IPW$$OF  |
 ---------+--------+-----+-+---+---------+-----------------------------------------------------------------+-+---------+
(MI)   TA |        |green| | - |$GAM D=L | 1Q6PI   POFFLOAD PICKUP PROCEEDING WITH mmmm OUT OF nnnn ..     |E|IPW$$OF  |
 ---------+--------+-----+-+---+---------+-----------------------------------------------------------------+-+---------+
 MI SP    |AK      |RED  | | - |$GAM D=L | 1Q6QI   JOB jobname jobno ENTRY KEPT IN CLASS 'A' WITH HOLD ..  |E|IPW$$AQ  |
 ---------+--------+-----+-+---+---------+-----------------------------------------------------------------+-+---------+
(MA) SP   |  (CM)  |RED  |x| - |$GAM+$WTO| 1Q6SA   TOO MANY CLASS ENTRIES FOUND - SURPLES IGNONRED         |E|IPW$$PS  |
 ---------+--------+-----+-+---+---------+-----------------------------------------------------------------+-+---------+
(MA) SP   |  (CM)  |RED  |x| - |$GAM+$WTO| 1Q6TA   DUPLICATE CLASS ENTIRES - FIRST ACCEPTED , DUPLICATES..|E|IPW$$PS  |
```

Figure 136 (Page 7 of 22). Message Reference

Routing Code RT= xx xx xx	Descrip- tor Code DC= xx xx xx	Color	C m n d	DOM 'ed	Issued via:	Message: (E) Module has IPW$GMM msg EQUATE $1xxxx ----------------+ (L) Module has locally defined message +--- Msg equate suffix $1xxx(n) if multiple messages v		Module: (where issued)
MA SP		RED		-	$GAM D=L	1Q6UA DBLK GROUP OWNERSHIP MISMATCH FOR QUEUE ENTRY ...	E	IPW$$GD
MA SP		RED		-	$GAM D=L	1Q6VA SEH=	E	IPW$$GD
(MI)*		green		-	$GAM D=L	1Q70I TASK FAILURE, STOPPED partition-id	E	IPW$$TR
(MI)*		green		-	$GAM+$WTO	1Q71I task, cuu TERMINATED	E	IPW$$TR
(MI)*		green		-	$GAM D=L	1Q72I PACCOUNT TERMINATED	E	IPW$$TR
(MI)*		green		-	$GAM+$WTO	1Q73I(0)STATUS DISPLAY TERMINATED	E	IPW$$TR
(MI)*		green		-	$GAM+$WTO	(1)TAPE STATUS DISPLAY TERMINATED	E	IPW$$TR
(MI)*		green		-	$GAM D=L	(2)STATUS DISPLAY TERMINATED (INCREASE DEFAULT ..	E	IPW$$TR
(MI)*SP	SF	RED		-	$GAM+$WTO	1Q74A ACCOUNT SUPPORT FUNCTIONS TERMINATED	E	IPW$$TR
							E	IPW$$RY
(MI)*		green		-	$GAM+$WTO	1Q75I MULTIPLE TERMINATION OF TASK, task,cuu TERMINATED	E	IPW$$TR
(MI)*SP	SF	RED		-	$GAM+$WTO	1Q76I VSE/POWER CANNOT CONTINUE, RC=nnnn	E	IPW$$TR
(MI) SP	SF	RED		-	$GAM D=L		E	IPW$$I3
(MI) SP	SF	RED		-			E	IPW$$I4
(or) TA	(CM)	green	x	-		1Q77I(0)INVALID ENTRY ON SPOOL TAPE ON dev FOR task,cuu, RC=n..	E	IPW$$CS
(MI) TA		green		-	$GAM D=L		E	IPW$$DM
(or) TA	(CM)	green	x	-	$GAM D=L		E	IPW$$PS
(MI) TA		green		-	$GAM D=L		E	IPW$$OF
(MI) TA		green		-	$GAM D=L		E	IPW$$OT
(MI) TA		green		-			E	IPW$$QM
(or) TA	(CM)	green	x	-	$GAM D=L	1Q77I(1)INVALID ENTRY ON SPOOL TAPE ON dev ..., SUGGEST SELECT.	E	IPW$$OF
MI SP	II	green		-		1Q78I(0)NO REAL/PFIXED STORAGE AVAILABLE FOR task,cuu	E	IPW$$AM
(or)SP	(CM)II	green	X	-			E	IPW$$CS
MI SP	II	green		-			E	IPW$$LM
MI SP	II	green		-			E	IPW$$SF
(or)SP	(CM)II	green	X	-		1Q78I(1)NO REAL/PFIXED STORAGE AVAILABLE FOR task,tapeaddr	E	IPW$$CPS
(MI)		green		-		1Q79I ACCOUNT FILE SAVED	E	IPW$$AM
(MI)		green		-			E	IPW$$SF
(or)SP	(CM)II	green	x	-		1Q7AI commandcode NO GETVIS-24 STORAGE AVAILABLE	E	IPW$$CB
(or)SP	(CM)II	green	x	-			E	IPW$$CD
(or)SP	(CM)II	green	x	-			E	IPW$$CF
(or)SP	(CM)II	green	x	-			E	IPW$$CG
(or)SP	(CM)II	green	x	-	$GAM D=L		E	IPW$$CLD
(or)SP(TA)	(CM)II	green	x	-			E	IPW$$CO
(or)SP	(CM)II	green	x	-			E	IPW$$CP
(or)SP	(CM)II	green	x	-			E	IPW$$CPS
(or)SP	(CM)II	green	x	-			E	IPW$$CS
(or)SP	(CM)II	green	x	-			E	IPW$$CT
(or)SP	(CM)II	green	x	-			E	IPW$$CU
(or)SP	(CM)II	green	x	-			E	IPW$$CX
(or)SP	(CM)II	green	X	-		1Q7BI commandcode NO REAL/PFIXED STORAGE AVAILABLE	E	IPW$$CD
(or)SP	(CM)II	green	X	-	$GAM D=L		E	IPW$$CLD
(MI)SP TA	II	green		-	$GAM D=L	1Q7CI TAPE SPOOLING FORCED TO SKIP FILE CLOSE DUE TO ...	E	IPW$$XWE
(MI) TA	II	green		-	$GAM D=L	1Q7DI(0)TAPE BEGINS WITH INCOMPLETE SPOOL ENTRY. SKIPPING ...	E	IPW$$NQ
(MI) TA	II	green		-	$GAM D=L	1Q7DI(1)TAPE BEGINS WITH INCOMPLETE SPOOL ENTRY. SKIPPING ...	E	IPW$$QM
(MI) TA	AK	RED		-	$GAM D=L	1Q7EA POFFLOAD SKIPPED ENTRY jobnm jobno qid DUE TO INSUFF..	E	IPW$$OF
(or)	(CM)	green	x	-	$GAM+$WTO	1Q7FI WRITER TASK REJECTED FOR 9346/3592 TAPE ON cuu, RC=n..	E	IPW$$CS
(MI)		green		-		1Q80I ACCOUNT FILE ERASED	E	IPW$$AM
(MI)		green		-			E	IPW$$SF
(MI)		green		-		1Q81I filename EXTENT TOO SMALL, COMMAND NOT EXECUTED	E	IPW$$AM
(MI)		green		-			E	IPW$$SF
(MI)		green		-		1Q82I PACCOUNT PROCESSING CANCELED BY COMMAND	E	IPW$$AM
(MI)		green		-			E	IPW$$SF

Figure 136 (Page 8 of 22). Message Reference

Routing Code RT= xx xx xx	Descrip-tor Code DC= xx xx xx	Color	Cmnd	DOM 'ed	Issued via:	Message: (E) Module has IPW$GMM msg EQUATE $1xxxx (L) Module has locally defined message +--- Msg equate suffix $1xxx(n) if multiple messages v		Module: (where issued)
(MI)*		green		-	$GAM D=L		E	IPW$$TR
(MI)		green		-		1Q83I ACCOUNT FILE NOTHING TO SAVE	E	IPW$$AM
(MI)		green		-			E	IPW$$SF
(MI)SP	SF	RED		-		1Q84I ACCOUNT INCOMPLETE FOR jobname jobnumber	E	IPW$$AM
(MI)SP	SF	RED		-			E	IPW$$PF
>VSE EXCP Default <		green		-	EXCP LOC	1Q85I task,cuu WAITING FOR GETVIS-24 STORAGE	L	IPW$$NU
MA DK	DA	WHITE		YES	$GAM D=L	1Q86A DISKETTE REQUIRED ON cuu, FOR jobname jobnumber, HDR1.	E	IPW$$ER
(MI)		green		-		1Q87I cuu EOJ ADDED FOR jobname jobnumber	E	IPW$$ER
(MI)		green		-			E	IPW$$SY
(MI)		green		-			E	IPW$$XTP
(MI)		green		-	$GAM D=L	1Q88I INVALID 3540 UNIT FOR partition-id cuu	E	IPW$$XRE
(MI)		green		-		1Q89I PROGRAM OUT OF SEQUENCE IN partition-id	E	IPW$$XRE
(or)	(CM)	green	x	-		1Q8AI TASK TRACE NOT YET STARTED	E	IPW$$CV
(or)	(CM)	green	x	-	$GAM D=L	1Q8BI STATISTICS STATUS REPORT DISPLAYED IN LIST ENTRY $STA.	E	IPW$$PS
(MI)		green		-		1Q8CI(0)DEFAULT OUTPUT VALUES USED FOR jobnam jobnu, ON nodeid	E	IPW$$XRE
(MI)		green		-		(1)DEFAULT OUTPUT VALUES USED FOR jobnam jobnu, SPOOLED .	E	IPW$$XRE
(or)	(CM)	green	x	-		1Q8DI INVALID CLASS class NOT ACCESSIBLE TO PVARY COMMAND	E	IPW$$CV
(or)	(CM)	green	x	-		1Q8EI ALL CLASSES FLAGGED INVALID IN ACTIVE DYNAMIC CLASS TBL	E	IPW$$CV
(MI)		green		-		1Q8FI VSE/SAM TAPE SPOOLING VIA SEGMENT MACRO PROHIBITED,...	E	IPW$$XJ
(or)	(CM)	green	x	-	$GAM+$WTO	1Q8GI STATUS REPORT DISPLAXED IN LIST ENTRY ... $DYNx.Z *	E	IPW$$PS
(or)	(CM)	green	x	-	$GAM+$WTO	1Q8HI(0)MESSAGE mmmmI BEEN ENABLED , NOW DISABBLED..$DYNx.Z *.	E	IPW$$CV
(or)	(CM)	green	x	-	$GAM+$WTO	1Q8HI(1)MESSAGE mmmmI BEEN ENABLED , NOW ENABLED ...$DYNx.Z *.	E	IPW$$CV
(or)	(CM)	green	x	-	$GAM+$WTO	1Q8HI(2)MESSAGE mmmmI BEEN ENABLED , NOW DISABBLED..$DYNx.Z *.	E	IPW$$CV
(or)	(CM)	green	x	-	$GAM+$WTO	1Q8HI(3)MESSAGE mmmmI BEEN ENABLED , NOW ENABLED ...$DYNx.Z *.	E	IPW$$CV
(or)	(CM)	green	x	-	$GAM+$WTO	1Q8HI(4)MESSAGE mmmmI IS DISABLED $DYNx.Z *	E	IPW$$CV
(or)	(CM)	green	x	-	$GAM+$WTO	1Q8JI(0)MESSAGE mmmmI NOT ACCEPTED $DYNx.Z *	E	IPW$$CV
(or)	(CM)	green	x	-	$GAM+$WTO	1Q8JI(1)MESSAGE mmmmI NOT VALID $DYNx.Z *	E	IPW$$CV
(or)	(CM)	green	x	-	$GAM+$WTO	1Q8JI(2)MESSAGE mmmmI CANNOT BE PROCESSED $DYNx.Z *	E	IPW$$CV
(MI)		green		-	$GAM D=L	1Q8KI(0)OUTPUT jobname jobnu PASSED TO PRINT ...	E	IPW$$LW
(MI)		green		-	$GAM D=L	1Q8KI(1)OUTPUT jobname jobnu PASSED TO PRINT ...	E	IPW$$LW
(MI)		green		-	$GAM D=L	1Q90I(0)* $$ RDR STATEMENT NOT ALLOWED, JOB FLUSHED	E	IPW$$LR
(MI) DK		green		-	$GAM D=L	(1)* $$ RDR STATEMENT NOT PROCESSED, JOB FLUSHED	E	IPW$$ER
(MA) DK	(DA)	WHITE		-	$WTR LOC	1Q91D cuu NON-COMPATIBLE DISKETTE FOR RDR,cuu2	L	IPW$$OE
(MA) DK	(DA)	WHITE		-	$WTR LOC	1Q91D VOL1 LABEL ERROR OR NOT FOUND R=	L	IPW$$OE
(MA) DK	(DA)	WHITE		-	$WTR LOC	1Q91D NON-BASIC EXCHANGE DISKETTE TYPE R=	L	IPW$$OE
(MA) DK	(DA)	WHITE		-	$WTR LOC	1Q91D NON-BASIC EXCHANGE FFFFFFFF FILE R=	L	IPW$$OE
(MA) DK	(DA)	WHITE		-	$WTR LOC	1Q91D FFFFFFFF BYPASS REQUIRED R=	L	IPW$$OE
(MA) DK	(DA)	WHITE		-	$WTR LOC	1Q91D LABEL STANDARD VERSION VIOLATION R=	L	IPW$$OE
(MA) DK	(DA)	WHITE		-	$WTR LOC	1Q91D MULTIVOLUME IND NOT C, L, OR BLANK R=	L	IPW$$OE
(MA) DK	(DA)	WHITE		-	$WTR LOC	1Q91D FFFFFFFF END XTNT BELOW BEGIN XTNT R=	L	IPW$$OE
(MA) DK	(DA)	WHITE		-	$WTR LOC	1Q91D FFFFFFFF EOD ADDR BELOW BEGIN XTNT R=	L	IPW$$OE
(MA) DK	(DA)	WHITE		-	$WTR LOC	1Q91D VOL SEQ NO.	L	IPW$$OE
(MA) DK	(DA)	WHITE		-	$WTR LOC	1Q91D BLOCKLENGTH	L	IPW$$OE
(MA) DK	(DA)	WHITE		-	$WTR LOC	1Q91D BEGINEXTENT ERR IN HDR1 LABEL (NNNNN) R=	L	IPW$$OE
(MA) DK	(DA)	WHITE		-	$WTR LOC	1Q91D END EXTENT	L	IPW$$OE
(MA) DK	(DA)	WHITE		-	$WTR LOC	1Q91D END-OF-DATA	L	IPW$$OE
(MA) DK	(DA)	WHITE		-	$WTR LOC	1Q92D cuu NO HDR1 FOR fileid, RDR,cuu2 R=	L	IPW$$OE
(MA) DK	(DA)	WHITE		-	$WTR LOC	1Q93D cuu SECURED VOLUME/FILE FOR RDR,cuu2 R=	L	IPW$$OE
(MA) DK	(DA)	WHITE		-	$WTR LOC	1Q94D cuu EXPECT VOL nn, NOT mm, RDR,cuu2 R=	L	IPW$$OE
(MA) DK	(DA)	WHITE		-	$WTR LOC	1Q95D cuu NON-VERIFIED fileid, RDR,cuu2 R=	L	IPW$$OE

Figure 136 (Page 9 of 22). Message Reference

Routing	Descrip-	Color	C	DOM	Issued	Message:		Module:
Code	tor Code		m	'ed	via:	(E) Module has IPW$GMM msg EQUATE $1xxxx ----------------	+	(where
RT=	DC=		n			(L) Module has locally defined message	|	issued)
			d			+--- Msg equate suffix $1xxx(n) if multiple messages	|	
xx xx xx	xx xx xx					v	v	
(MI) DK		green		-	$WTO LOC	1Q96I cuu fileid IS EMPTY FILE FOR RDR,cuu2	L	IPW$$OE
(MA) DK	(DA)	WHITE		-	$WTR LOC	1Q97D cuu PREMATURE LAST VOL FOR RDR,cuu2 R=	L	IPW$$OE
(MA) DK	(DA)	WHITE		-	$WTR LOC	1Q98D cuu fileid TOO MANY VOLS RDR,cuu2 R=	L	IPW$$OE
(MA) DK	(DA)	WHITE		-	$WTR LOC	1Q9nD INVALID RESPONSE R=	L	IPW$$OE
(MA) DK	(DA)	WHITE		-	$WTR LOC	1Q9nD NO PRECEEDING VOL, INCONSIST RESP R=	L	IPW$$OE
(MI)SP	(CM)	green		-	$GAM+$WTO	1Q9GI BIGGEST SORTED DISPLAYED IN LIST ENTRY ...	E	IPW$$PS
(MI)SP	SF	RED		-		1QA0I NO SUBTASK AVAILABLE FOR task,cuu	E	IPW$$AS
(MI)SP	SF	RED		-			E	IPW$$LD4
(MI)SP	SF	RED		-	$GAM D=L		E	IPW$$TI
(MI)SP	SF	RED		-		1QA1I SETPRT ROUTINE NOT FOUND IN SVA task,cuu	E	IPW$$AS
(MI) TA		green		-	$GAM D=L	1QA2I VSE/POWER MULTI-VOLUME TAPE COMPLETE FOR jobname ...	E	IPW$$OF
(MI) TA		green		-	$GAM D=L		E	IPW$$OT
(MI)SP	II	green		-		1QA3I SETPRT ERROR FOR jobname jobnumber task,cuu	E	IPW$$AS
(MI)SP	II	green		-		1QA4I OUTPUT PROCESSING STOPPED FOR jobname jobnumber task	E	I PW$$AS
MA	DA	WHITE		Yes		1QA5A cuu SETUP REQUIRED jobname FORM=ffff FLASH=hhhh THREAD.	E	IPW$$LW
MI SP	II	green		-		1QA6I(0)NO STORAGE AVAILABLE FOR ttttt, cuu	E	IPW$$AS
(MI)SP	SF	RED		-	$GAM D=L	(1)	E	IPW$$I7
MA	DA	WHITE		Yes	$GAM D=L	1QA7A MOUNT TRAIN FOR UCS=uuuuuuuu jobname jobnumber task,cuu	E	IPW$$PL
(MI)		green		-	$GAM D=L	1QA8I ON cuu BAND bandid NEEDED FOR jobname jobnumber	E	IPW$$PL
MA	DA	WHITE		Yes		1QA9A task,cuu WAITING FOR OPERATOR REACTIVATION	E	IPW$$LW
(MI)		green		-	$GAM D=L	1QAAI USERID userid UNKNOWN TO VM, ...	E	IPW$$LW
(or)	(CM)	green		-	$GAM+$WTO	1QABI TASK task,tcuu ACTIVE USING cuu, COMMAND IGNORED	E	IPW$$CP
(or)	(CM)	green		-	$GAM+$WTO	1QACI cuu IS NOT ASSIGNED TO VSE/POWER, COMMAND IGNORED	E	IPW$$CP
(or)	(CM)	green		-	$GAM+$WTO	1QADI cuu IS NEITHER A PRINT NOR A PUNCH ... COMMAND IGNORED	E	IPW$$CP
(or)	(CM)	green		-	$GAM+$WTO	1QAEI TASK task,cuu IN STATE WHERE ... COMMAND IGNORED	E	IPW$$CP
(MI)		green		-	$GAM D=L	1QAFI SHARING SYSTEM SYSID=xxxxxxxx ...,COMMAND IGNORED	E	IPW$$I3
(or)	(CM)	green	x	-	$GAM+$WTO	1QAGI PSTOP 'DBLKTR' OBSOLETE -	E	IPW$$CP
(MA)		WHITE		-	$GAM D=L	1QAFD IF SYSID=xxxxxxxx CURRENTLY INACTIVE, ALLOW WARM START?	E	IPW$$I3
(MI)SP	SF	RED		-	$GAM D=L	1QB0I SUPERVISOR WITHOUT DASD SHARING FEATURE	E	IPW$$I2
(MI)SP	SF	RED		-	$GAM D=L	1QB1I filename NOT ON SHARED DEVICE	E	IPW$$I2
(MI)SP	SF	RED		-			E	IPW$$I4
(MA)	(DA)	WHITE		-	$GAM+$WTR	1QB2D IS ANY OTHER VSE/POWER SYSTEM ALREADY INITIALIZED ?	E	IPW$$I2
(MI)		green		-	$WTO LOC	1QB3A(0)SHARED PHASE=pppppppp REQUESTING WARM START. .	E	IPW$$I3
(MI)		green		-	$WTO LOC	(1)NON SHARED PHASE=pppppppp REQZESTUBG WARN ST A.	E	IPW$$I3
(MA)	(DA)	WHITE		-	$GAM+$WTR	1QB3D(0)IF SWITCH FROM NON SHARED TO SHARED PROCESS. .	E	IPW$$I3
(MA)	(DA)	WHITE		-	$GAM+$WTR	(1)IF SWITCH FROM SHARED TO NON SHARED PROCESS. .	E	IPW$$I3
>VSE EXCP Default <		green		-	EXCP REQ	1QB4I LOCK TABLE SPACE EXHAUSTED	E	IPW$$TI
(MI) SP	SF	RED		-	$GAM D=L	1QB5I INTERNAL MACRO CALL FAILED IN PHASE=xxxxxxxx, RC=rrmm	E	IPW$$AS
(MI) SP	SF	RED		-	$GAM D=L		E	IPW$$CLD
(MI) SP	SF	RED		-	$GAM D=L		E	IPW$$CS
(MI) SP	SF	RED		-	$GAM D=L		E	IPW$$CV
(MI) SP	SF	RED		-	$GAM D=L		E	IPW$$DP
(MI) SP	SF	RED		-	$GAM D=L		E	IPW$$IP
(MI) SP	SF	RED		-	$GAM D=L		E	IPW$$I2

Figure 136 (Page 10 of 22). Message Reference

```
  Routing    |Descrip-|Color|C|DOM|Issued   |Message:                                                          | Module: |
  Code       |tor Code|     |m|'ed| via:    |       (E) Module has IPW$GMM msg EQUATE  $1xxxx ----------------+|(where   |
  RT=        |DC=     |     |n|   |         |       (L) Module has locally defined message                    || issued) |
             |        |     |d|   |         |       +--- Msg equate suffix  $1xxx(n) if multiple messages     ||         |
  xx xx xx   |xx xx xx|     | |   |         |       v                                                        v|         |

 (MI) SP     |SF      |RED  | | - |$GAM D=L |                                                                |E|IPW$$I3  |
 (MI) SP     |SF      |RED  | | - |         |                                                                |E|IPW$$I4  |
 (MI) SP     |SF      |RED  | | - |         |                                                                |E|IPW$$I5  |
 (MI) SP     |SF      |RED  | | - |$GAM D=L |                                                                |E|IPW$$I7  |
 (MI) SP     |SF      |RED  | | - |$GAM D=L |                                                                |E|IPW$$LD4 |DY42713
 (MI) SP     |SF      |RED  | | - |         |                                                                |E|IPW$$LU  |
 (MI) SP     |SF      |RED  | | - |         |                                                                |E|IPW$$LW  |
 (MI) SP     |SF      |RED  | | - |         |                                                                |E|IPW$$NU  |
 (MI) SP     |SF      |RED  | | - |         |                                                                |E|IPW$$SM  |
 (MI)*SP     |SF      |RED  | | - |$GAM D=L |                                                                |E|IPW$$TR  |
 (MI) SP     |SF      |RED  | | - |$GAM D=L |                                                                |E|IPW$$T1  |
 (MI) SP     |SF      |RED  | | - |         |                                                                |E|IPW$$XRE |
 (MI) SP     |SF      |RED  | | - |         |                                                                |E|IPW$$XWE |
 >VSE Exception Msg< |RED  | | - |EXCP LOC |        INTERNAL MACRO CALL FAILED IN PHASE=IPW$$AT, RC=rr26    |L|IPW$$AT  |
 >VSE Excpetion Msg< |RED  | | - |EXCP LOC |        INTERNAL MACRO CALL FAILED IN PHASE=IPW$$CM, RC=rr28    |L|IPW$$CM  |
 >VSE Excpetion Msg< |RED  | | - |EXCP LOC |        INTERNAL MACRO CALL FAILED IN PHASE=IPW$$IP, RC=rr27    |L|IPW$$IP  |
 >VSE Exception Msg< |RED  | | - |EXCP LOC |        INTERNAL MACRO CALL FAILED IN PHASE=IPW$$MS, RC=rrmm    |L|IPW$$MS  |
 >VSE Exception Msg< |RED  | | - |EXCP REQ |        INTERNAL MACRO CALL FAILED IN PHASE=IPW$$TI, RC=rrmm    |E|IPW$$TI  |
  ----------+--------+-----+-+---+---------+----------------------------------------------------------------+-+---------+
 >VSE EXCP Default < |green| | - |EXCP REQ | 1QB6I   QUEUE FILE LOCKED BY ANOTHER SYSTEM                    |E|IPW$$TI  |
 (MI)        |        |green| | - |$GAM D=L |                                                                |E|IPW$$I2  |
  ----------+--------+-----+-+---+---------+----------------------------------------------------------------+-+---------+

                     .* 6.6 rc 6 on
 (MI)        |        |green| | - |$GAM+$WTO| 1QB7I   PARTIAL QUEUE FILE RECOVERY IN PROGRESS <FOR SYSID n1..|E|IPW$$RY  |
                     .* 6.6 rc 6 off
  ----------+--------+-----+-+---+---------+----------------------------------------------------------------+-+---------+
 (MI)        |        |green| | - |$GAM D=L | 1QB8I   QUEUE FILE RECOVERY COMPLETED                          |E|IPW$$RY  |
 (MI)        |        |green| | - |         |                                                                |E|IPW$$QM  |
  ----------+--------+-----+-+---+---------+----------------------------------------------------------------+-+---------+
  MA    TA   |   DA   |WHITE| |Yes|$WTO LOC | 1QB9A   cuu, HEADER= filelabel creation date                   |L|IPW$$OT  |
  ----------+--------+-----+-+---+---------+----------------------------------------------------------------+-+---------+
 (MI)SP      |SF      |RED  | | - |         | 1QBAI   QUEUE FILE RECOVERY IN PROGRESS FOR FREE QUEUE RECORD. |E|IPW$$RQ  |
  ----------+--------+-----+-+---+---------+----------------------------------------------------------------+-+---------+
 (MI)   TA   |        |green| | - |$GAM D=L |  1QBBI  RESTART/SETUP OF SPOOL TAPE PROCESSING REQUESTED AT .. |E|IPW$$LW  |
  ----------+--------+-----+-+---+---------+----------------------------------------------------------------+-+---------+
 (MI)SP      |      II|green| | - |$GAM D=L | 1QBCI   QUEUE FILE RECOVERY DETECTED NEW DISP=X JOB(S) IN...   |E|IPW$$RY  |
  ----------+--------+-----+-+---+---------+----------------------------------------------------------------+-+---------+
 >VSE EXCP Default < |green| | - |EXCP REQ | 1QBDI   PREVIOUS CONSOLE DISPLAY MESSAGE(S) HAS BEEN LOST ...  |E|IPW$$MS  |
  ----------+--------+-----+-+---+---------+----------------------------------------------------------------+-+---------+
 (MI)SP      |SF      |RED  | | - |$GAM D=L |  1QBEI  INTERNAL MACRP CALL 'CPCOM' FAILED IN PHASE= ...       |E|IPW$$LW  |
 (MI)SP      |SF      |RED  | | - |$GAM D=L |                                                                |E|IPW$$TR  |
  ----------+--------+-----+-+---+---------+----------------------------------------------------------------+-+---------+
 (MI)SP      |II      |RED  | | - |$GAM D=L |  1QBFI  $IJBXPCA ERROR FOR PARTITION pid ...                   |E|IPW$$AS  |
  ----------+--------+-----+-+---+---------+----------------------------------------------------------------+-+---------+
 (MA)        |  (DA)  |WHITE| | - |$GAM+$WTR| 1QBGD(0)NON SHARED VSE/POWER SYSTEM FOUND. IF STILL          ..|E|IPW$$I3  |
 (MA)        |  (DA)  |WHITE| | - |$GAM+$WTR|      (1)SHARED VSE/POWER SYSID(S)=nnnnnnnn FOUND -           ..|E|IPW$$I3  |
  ----------+--------+-----+-+---+---------+----------------------------------------------------------------+-+---------+
 (MI)SP      |        |green| | - |         | 1QC0I   SLI STATEMENT REJECTED, JOB jobname jobnum FLUSHED.... |E|IPW$$SL  |
  ----------+--------+-----+-+---+---------+----------------------------------------------------------------+-+---------+
 (MI)SP      |        |green| | - |         | 1QC1I   UNABLE TO PROCESS MEMBER member.tp JOB FLUSHED RC=..   |E|IPW$$AS  |
 (MI)SP SE   |        |green| | - |         | 1QC1I   UNABLE TO PROCESS MEMBER member.tp JOB FLUSHED RC=0004 |E|IPW$$AS  |
             |        |     | |   |         |      ==>  (RC=0004 will be captured in IPW$$MS)                | |         |
  ----------+--------+-----+-+---+---------+----------------------------------------------------------------+-+---------+
 (MI)        |        |green| | - |         | 1QC2I   SLI NESTING ERROR FOR MEMBER member.type, JOB......... |E|IPW$$SL  |
  ----------+--------+-----+-+---+---------+----------------------------------------------------------------+-+---------+
 (MI)        |        |green| | - |         | 1QC3I(0)MEMBER member.type NOT FOUND, JOB jobname jobnum...... |E|IPW$$AS  |
 (MI)        |        |green| | - |         |      (1)MEMBER member.type NOT FOUND IN lib.sublib,JOB jobnam. |E|IPW$$AS  |
  ----------+--------+-----+-+---+---------+----------------------------------------------------------------+-+---------+
 (MI)SP      |SF      |RED  | | - |         | 1QC4I   macroname MACRO FAILED, RC/FDK=nn,nn JOB jobname...... |E|IPW$$AS  |
  ----------+--------+-----+-+---+---------+----------------------------------------------------------------+-+---------+
 >VSE EXCP Default < |WHITE| | - |EXCP LOC | 1QC5D   TO DUMP TO PRINTER OR TAPE, SPECIFY ...                |L|IPW$$AT  |
 >VSE EXCP Default < |WHITE| | - |EXCP LOC |         PRINTER/TAPE TYPE INVALID OR NOT FREE ...              |L|IPW$$AT  |
  ----------+--------+-----+-+---+---------+----------------------------------------------------------------+-+---------+
 >VSE EXCP Default < |green| | - |EXCP LOC | 1QC5I   DUMP PROCESSING FAILED, RC=NN                          |L|IPW$$ID  |
  ----------+--------+-----+-+---+---------+----------------------------------------------------------------+-+---------+
 (MI)        |        |green| | - |         | 1QC6I   LIBRARY libname NOT FOUND, JOB jobname jobnumb FLUSHED |E|IPW$$AS  |
  ----------+--------+-----+-+---+---------+----------------------------------------------------------------+-+---------+
 (MI)        |        |green| | - |         | 1QC7I   part-id jobname jobnumber FROM ... COMPLETE ...        |E|IPW$$XRE |
  ----------+--------+-----+-+---+---------+----------------------------------------------------------------+-+---------+
  MA         |   DA   |WHITE| | - |$GAM D=L | 1QD1A   TOO MANY EXTENTS (mm) FOR DATA FILE EXTENSION RC=..... |E|IPW$$I4  |
  ----------+--------+-----+-+---+---------+----------------------------------------------------------------+-+---------+
 (MI)SP      |SF      |RED  | | - |$GAM+$WTR| 1QD2D   DATA FILE EXTEND NO. nn - FOR FORMATTING REPN RC=....Y.|E|IPW$$I4  |
 (MI)        |        |green| | - |$GAM D=L | 1QD2I   EXISTING DATE FILE EXTENT NO. nn FOUND IN ..N RC=......|E|IPW$$I4  |
```

Figure 136 (Page 11 of 22). Message Reference

```
 Routing    |Descrip-|Color|C|DOM|Issued   |Message:                                                        | Module: |
 Code       |tor Code|     |m|'ed| via:    |       (E) Module has IPW$GMM msg EQUATE  $1xxxx ----------------+|(where   |
 RT=        |DC=     |     |n|   |         |       (L) Module has locally defined message                    || issued) |
            |        |     |d|   |         |       +--- Msg equate suffix  $1xxx(n) if multiple messages     ||         |
 xx xx xx   |xx xx xx|     | |   |         |       v                                                         v|         |

 ----------+--------+-----+-+---+---------+-----------------------------------------------------------------+-+---------+
 MA        |   DA   |WHITE| | - |$GAM D=L | 1QD3A   DATA FILE EXTENSION FAILED FOR EXTENT mm RC= ...       |E|IPW$$I4  |
 ----------+--------+-----+-+---+---------+-----------------------------------------------------------------+-+---------+
(MI)       |        |green| | - |$GAM D=L | 1QD4I   VERIFYING LOCATION OF ADDITIONAL DATA FILE EXTENT mm    |E|IPW$$I4  |
 ----------+--------+-----+-+---+---------+-----------------------------------------------------------------+-+---------+
(MI)       |        |green| | - |$GAM D=L | 1QD5I   LOCATION OF ADDITIONAL DATA FILE EXTENT mm VERIFIED SUC|E|IPW$$I4  |
 ----------+--------+-----+-+---+---------+-----------------------------------------------------------------+-+---------+
    SP     |SF      |RED  | |Yes|$GAM D=L | 1QD6I(0)FORMATTING OF NEW DATA FILE EXTENT NO: nn ST STARTED AR|E|IPW$$T1  |
    SP     |SF      |RED  | |Yes|$GAM D=L | 1QD6I(1)FORMATTING OF NEW DATA FILE EXTENT NO: nn CO STARTED MP|E|IPW$$T1  |
    SP     |SF      |RED  | |Yes|$GAM D=L | 1QD6I(2)FORMATTING OF NEW DATA FILE EXTENT NO: nn DE STARTED CT|E|IPW$$T1  |
    SP     |SF      |RED  | | - |$GAM D=L | 1QD6I(3)FORMATTING OF NEW DATA FILE EXTENT NO: nn FA FAILED .IL|E|IPW$$T1  |
    SP     |SF      |RED  | | - |$GAM D=L | 1QD6I(4)FORMATTING OF NEW DATA FILE EXTENT NO: nn PO FAILED .ST|E|IPW$$T1  |
 ----------+--------+-----+-+---+---------+-----------------------------------------------------------------+-+---------+
 MA        |   DA   |WHITE| |Yes|$GAM D=L | 1QD7A   ADDITIONAL EXTENT(S) FOUND FOR EXTENSION OF  FAILED ...|E|IPW$$I4  |
 ----------+--------+-----+-+---+---------+-----------------------------------------------------------------+-+---------+
(MI)       |        |green| |   |$GAM D=L | 1QE1I   RE-ALLOCATION PROCESS STARTED FOR VSE/POWER QUEUE FILE |E|IPW$$I3  |
 ----------+--------+-----+-+---+---------+-----------------------------------------------------------------+-+---------+
 MA        |   DA   |WHITE| |   |$GAM D=L | 1QE2A(0)RE-ALLOCATION OF QUEUE FILE FAILED, RC=...CONTINUED     |+|IPW$$I3  |
 MA        |   DA   |WHITE| |   |$GAM D=L | 1QE2A(1)RE-ALLOCATION OF QUEUE FILE FAILED, RC=...TERMINATED    |+|IPW$$I3  |
 ----------+--------+-----+-+---+---------+-----------------------------------------------------------------+-+---------+
(MI)SP     |SF      |RED  | |   |$GAM+$WTR| 1QE3D   CONFIRM QUEUE FILE RE-ALLOCATION FROM IJQFOLD TO ....   |E|IPW$$I3  |
 ----------+--------+-----+-+---+---------+-----------------------------------------------------------------+-+---------+
(MI)       |        |green| |   |$GAM D=L | 1QE3I(0)IJQFOLD: // EXTENT SYS034, ......                       |E|IPW$$I3  |
(MI)       |        |green| |   |$GAM D=L | 1QE3I(1)IJQFILE: // EXTENT SYS034, ......                       |E|IPW$$I3  |
 ----------+--------+-----+-+---+---------+-----------------------------------------------------------------+-+---------+
(MI)       |        |green| |   |$GAM D=L | 1QE4I   VERIFYING LOCATION OF NEW QUEUE FILE IJQFILE BY OPEN ..|E|IPW$$I3  |
 ----------+--------+-----+-+---+---------+-----------------------------------------------------------------+-+---------+
(MI)       |        |green| |   |$GAM D=L | 1QE5I   LOCATION OF NEW QUEUE FILE IJQFILE VERIFIED SUCCESSFUL |E|IPW$$I3  |
 ----------+--------+-----+-+---+---------+-----------------------------------------------------------------+-+---------+
 MA SP     |   DA   |WHITE| |   |$GAM D=L | 1QE6A   CONFIRM QUEUE FILE RE-ALLOCATION FROM IJQFOLD TO ....   |E|IPW$$I3  |
 ----------+--------+-----+-+---+---------+-----------------------------------------------------------------+-+---------+
(MI)SP     |SF      |RED  | |   |$GAM D=L | 1QE7I   DELETION OF IJQFILE FAILED, REMOVE FILE-ID              |E|IPW$$I3  |
 ----------+--------+-----+-+---+---------+-----------------------------------------------------------------+-+---------+
 MA SP     |   DA   |WHITE| |   |$GAM D=L | 1QE8A   IJQFILE (// EXTENT SYS001 ... MISMATCH WITH ...         |E|IPW$$I3  |
 ----------+--------+-----+-+---+---------+-----------------------------------------------------------------+-+---------+
(MI)SP     |      II|green| | - |         | 1QF0I   DATA FILE nnn% FULL - QUEUE FILE nnn% FULL              |E|IPW$$QM  |
 ----------+--------+-----+-+---+---------+-----------------------------------------------------------------+-+---------+
(MI)SP     |SF      |RED  | | - |$GAM D=L | 1QF1I   UNABLE TO PLACE ENTIRE QUEUE FILE IN STORAGE, nnnnn K .|E|IPW$$I3  |
 ----------+--------+-----+-+---+---------+-----------------------------------------------------------------+-+---------+
 MA SP(MI)*|AK      |RED  | | - |$GAM+$WTO| 1QF2A   PEND FORCE REQUIRED TO TERMINATE VSE/POWER              |E|IPW$$TR  |
 ----------+--------+-----+-+---+---------+-----------------------------------------------------------------+-+---------+
(MI)SP     |SF      |RED  | | - |$GAM D=L | 1QF3I   VSE/POWER CONTINUES WITH A SUBSET OF QUEUE FILE - QUEUE|E|IPW$$I3  |
 ----------+--------+-----+-+---+---------+-----------------------------------------------------------------+-+---------+
 MI SP     |      II|green| | - |         | 1QF4I   NO FREE QUEUE RECORD AVAILABLE FOR task,cuu             |E|IPW$$QM  |
 ----------+--------+-----+-+---+---------+-----------------------------------------------------------------+-+---------+
(MI)*SP    |      II|green| | - |$GAM+$WTO| 1QF5I   QUEUE FILE IS BEING RE-BUILT                            |E|IPW$$TR  |
 ----------+--------+-----+-+---+---------+-----------------------------------------------------------------+-+---------+
(MI)*SP    |      II|green| | - |$GAM+$WTO| 1QF6I   QUEUE FILE SUCCESSFULLY RE-BUILT                        |E|IPW$$TR  |
 ----------+--------+-----+-+---+---------+-----------------------------------------------------------------+-+---------+
(MI)*SP    |AK      |RED  | | - |$GAM+$WTO| 1QF7A   QUEUE FILE DAMAGED - COLD START REQUIRED AFTER ....... |E|IPW$$TR  |
 ----------+--------+-----+-+---+---------+-----------------------------------------------------------------+-+---------+
(MI)SP     |SF      |RED  | | - |$GAM D=L | 1QF8I(0)nnnn FREE DBLK GROUPS (ABOUT xx%) LOST                  |E|IPW$$QM  |
(MI)SP     |SF      |RED  | | - |$GAM D=L | 1QF8I(1)nnnn FREE DBLK GROUPS OF A SUBCHAIN (ABOUT .          . |E|IPW$$QM  |
(MI)*SP    |SF      |RED  | | - |$GAM+$WTO|                                                                 |E|IPW$$TR  |
 ----------+--------+-----+-+---+---------+-----------------------------------------------------------------+-+---------+
(MA)       | (DA)   |WHITE| | - |$GAM+$WTR| 1QF9D   ANY OTHER VSE/POWER SYSTEM STILL RUNNING ? (REPLY: ... |E|IPW$$I3  |
 ----------+--------+-----+-+---+---------+-----------------------------------------------------------------+-+---------+
 MA SP     |AK      |RED  | | - |$GAM D=L | 1QFAA   USED DBLK GROUP FOUND IN A FREE DBLK GROUP SUBCHAIN     |E|IPW$$Q1  |
 ----------+--------+-----+-+---+---------+-----------------------------------------------------------------+-+---------+
 MA SP     |AK      |RED  | | - |$GAM D=L | 1QFBA   FREE DBLK GROUP FOUND IN RETURNED QUEUE ENTRY           |E|IPW$$Q1  |
 ----------+--------+-----+-+---+---------+-----------------------------------------------------------------+-+---------+
 MA SP     |AK      |RED  | | - |$GAM D=L | 1QFCA   MISMATCH OF GROUP COUNT AND ACTUAL NUMBER OF DBLK GROUP|E|IPW$$Q1  |
 ----------+--------+-----+-+---+---------+-----------------------------------------------------------------+-+---------+
 MA SP     |AK      |RED  | | - |$GAM D=L | 1QFDA   MISMATCH OF SUBCHAIN COUND AND ACTUAL NUMBER OF FREE GP|E|IPW$$Q1  |
 ----------+--------+-----+-+---+---------+-----------------------------------------------------------------+-+---------+
(MA)SP     | (DA)   |WHITE| | - |$WTR LOC | 1QFED  VSE/POWER GEN SECNODE VALUE xxxx DOESN'T MATCH ........ |L|IPW$$I3  |
 ----------+--------+-----+-+---+---------+-----------------------------------------------------------------+-+---------+
(MA)SP     | (DA)   |WHITE| | - |$WTR LOC | 1QFFD  VSE/POWER WARMSTART AND VSE ACCESS CONTROL NOT ACTIVATED|L|IPW$$I3  |
 ----------+--------+-----+-+---+---------+-----------------------------------------------------------------+-+---------+
 ----------+--------+-----+-+---+---------+-----------------------------------------------------------------+-+---------+
 MA    TA  |   DA   |WHITE| | - |$GAM D=L | 1QG0A(0)WRONG SPOOL TAPE MOUNTED VOLUME=nnn, PLEASE MOUNT  ... |E|IPW$$OT  |
 MA    TA  |   DA   |WHITE| | - |$GAM D=L | 1QG0A(1)WRONG SPOOL TAPE MOUNTED RC=nnn,     PLEASE MOUNT  ... |E|IPW$$OT  |
 ----------+--------+-----+-+---+---------+-----------------------------------------------------------------+-+---------+
(MI)       |      II|green| | - |$GAM D=L | 1QH0I   RE-ATTEMPT WARMSTART WHEN NO FURTHER VSE/PWR SYSTEM UP |E|IPW$$I3  |
```

Figure 136 (Page 12 of 22). Message Reference

```
 Routing   |Descrip-|Color|C|DOM|Issued   |Message:                                                           | Module: |
 Code      |tor Code|     |m|'ed| via:    |       (E) Module has IPW$GMM msg EQUATE  $1xxxx -----------------+|(where   |
 RT=       |DC=     |     |n|   |         |       (L) Module has locally defined message                     ||issued)  |
           |        |     |d|   |         |       +--- Msg equate suffix  $1xxx(n) if multiple messages      ||         |
 xx xx xx  |xx xx xx|     | |   |         |       v                                                          v|         |

 ----------+--------+-----+-+---+---------+-----------------------------------------------------------------+-+---------+
(MA)       |  (DA)  |WHITE| | - |$GAM+$WTR| 1QH1D   COLDSTART REQUESTED BY ANY SHARED SYSTEM ? (REPLY: ... |E|IPW$$I3  |
 ----------+--------+-----+-+---+---------+-----------------------------------------------------------------+-+---------+
(MI)SP     |SF      |RED  | | - |$GAM D=L | 1QH2I   IMMEDIATE TERMINATION ENTERED FOR SYSID x, RC=nnnn     |E|IPW$$TI  |
(MI)*SP    |SF      |RED  | | - |$GAM+$WTO|                                                                 |E|IPW$$TR  |
 ----------+--------+-----+-+---+---------+-----------------------------------------------------------------+-+---------+
(MI)SP     |SF      |RED  | | - |$GAM D=L | 1QH3I   nnnnn OF mmmmmm DBLK GROUPS LOST                        |E|IPW$$RY  |
 ----------+--------+-----+-+---+---------+-----------------------------------------------------------------+-+---------+
(MI)*SP    |SF      |RED  | | - |$GAM+$WTO| 1QH4I   CHECKPOINT OPTION WITHDRAWN FOR jobname jobnumber       |E|IPW$$TR  |
 ----------+--------+-----+-+---+---------+-----------------------------------------------------------------+-+---------+
(MI)       |     II |green| | - |$GAM D=L | 1QH5I   ENTERING QUEUE FILE REPAIR PHASE, TIME=.....            |E|IPW$$TI  |
 ----------+--------+-----+-+---+---------+-----------------------------------------------------------------+-+---------+
(MI)       |     II |green| | - |$GAM D=L | 1QH6I   SUCCESSFUL EXIT OF QUEUE FILE REPAIR PHASE              |E|IPW$$TI  |
 ----------+--------+-----+-+---+---------+-----------------------------------------------------------------+-+---------+
 MA SP     |AK      |RED  | | - |$GAM D=L | 1QH7A   REAL/PFIXED STORATE CORRUPTED - SHUTDOWN SYSTEM AND RE.|E|IPW$$NU  |
 ----------+--------+-----+-+---+---------+-----------------------------------------------------------------+-+---------+
>Dummy: Not Printed<|     |x| - |$GAM D=L | 1QLSC   >... last connected command response message ...<       |E|IPW$$CM  |
>Dummy: Not Printed<|     |x| - |$GAM D=L |                                                                 |E|IPW$$PS  |
>Dummy: Not Printed<|     |x| - |$GAM D=L |                                                                 |E|IPW$$TR  |
 ----------+--------+-----+-+---+---------+-----------------------------------------------------------------+-+---------+
           |        |     | |   |         | 1QTxx   >> Internal Trace Facility messages <<                  | |         |
           |        |     | |   |         |              (See Trace Manual)                                 | |         |
 ----------+--------+-----+-+---+---------+-----------------------------------------------------------------+-+---------+
(MI)SP     |SF      |RED  | | - |         | 1QX1I   XPCC FUNC=ffffffff FAILED IN PHASE=xxxxxxxx, RC=nn ... |E|IPW$$NS  |
(MI)SP     |SF      |RED  | | - |         |                                                                 |E|IPW$$XM  |
(MI)SP     |SF      |RED  | | - |         |                                                                 |E|IPW$$XT  |
(MI)SP     |SF      |RED  | | - |         |                                                                 |E|IPW$$XTS |
(MI)SP     |SF      |RED  | | - |         |                                                                 |E|IPW$$XH  |
 ----------+--------+-----+-+---+---------+-----------------------------------------------------------------+-+---------+
(MI)SP     |SF      |RED  | | - |         | 1QX2I   UNABLE TO CONTINUE CROSS-PARTITION SUPPORT              |E|IPW$$XM  |
 ----------+--------+-----+-+---+---------+-----------------------------------------------------------------+-+---------+
(MI)       |        |green| | - |         | 1QX3I   CROSS-PARTITION TASK taskid SERVING SAS=applid STOPPED |E|IPW$$XT  |
 ----------+--------+-----+-+---+---------+-----------------------------------------------------------------+-+---------+
(MI)       |        |green| | - |         | 1QY0I   START-UP FOR DEVICE devname UNSUCCESSFUL, DDS=ddsname  |E|IPW$$XT  |
 ----------+--------+-----+-+---+---------+-----------------------------------------------------------------+-+---------+
(MI)       |        |green| | - |         | 1QY1I   DEVICE devname UNAVAILABLE, DDS=ddsname, RC=xxxx       |E|IPW$$XTS |
 ----------+--------+-----+-+---+---------+-----------------------------------------------------------------+-+---------+
(MI)       |        |green| | - |         | 1QY2I   DEVICE devname WAITING FOR WORK, DDS=ddsname            |E|IPW$$XTG |
 ----------+--------+-----+-+---+---------+-----------------------------------------------------------------+-+---------+
(MI)       |        |green| | - |         | 1QY3I   DEVICE devname STARTED, DDS=ddsname, TIME=.....         |E|IPW$$XTS |
 ----------+--------+-----+-+---+---------+-----------------------------------------------------------------+-+---------+
(MI)       |        |green| | - |         | 1QY4I(0)DEVICE devname STOPPED BY OPERATOR userid, DDS=ddsname |E|IPW$$XT  |
(MI)SP     |SF      |RED  | | - |         |      (1)DEVICE devname STOPPED BY VSE/POWER, DDS=ddsname       |E|IPW$$XT  |
 ----------+--------+-----+-+---+---------+-----------------------------------------------------------------+-+---------+
(MI)       |        |green| | - |         | 1QY5I   TERMINATION OF DDS ddsname FOR DEVICE devname, RC=xxxx |E|IPW$$XT  |
 ----------+--------+-----+-+---+---------+-----------------------------------------------------------------+-+---------+
(MI)       |        |green| | - |         | 1QY6I   commandcode COMMAND NOT ACCEPTED BY DDS ddsname, RC=.. |E|IPW$$XTS |
 ----------+--------+-----+-+---+---------+-----------------------------------------------------------------+-+---------+
(or)       |  (CM)  |green|x| - |         | 1QY7I   DEVICE devname ALREADY STARTED                          |E|IPW$$CS  |
 ----------+--------+-----+-+---+---------+-----------------------------------------------------------------+-+---------+
(or)       |  (CM)  |green|x| - |         | 1QY8I   DEVICE devname UNKNOWN OR NOT YET STARTED               |E|IPW$$CF  |
(or)       |  (CM)  |green|x| - |         |                                                                 |E|IPW$$CG  |
(or)       |  (CM)  |green|x| - |         |                                                                 |E|IPW$$CI  |
(or)       |  (CM)  |green|x| - |         |                                                                 |E|IPW$$CP  |
(or)       |  (CM)  |green|x| - |         |                                                                 |E|IPW$$CT  |
(or)       |  (CM)  |green|x| - |         |                                                                 |E|IPW$$CU  |
(or)       |  (CM)  |green|x| - |         |                                                                 |E|IPW$$CX  |
 ----------+--------+-----+-+---+---------+-----------------------------------------------------------------+-+---------+
(MI)       |        |green| | - |         | 1QY9I   UNABLE TO START DEVICE devname, DDS=ddsname IN SHUTDOWN|E|IPW$$XT  |
 ----------+--------+-----+-+---+---------+-----------------------------------------------------------------+-+---------+
(MI)SP -PG |SF      |RED  | | - |$GAM D=L | 1QZ0I   SEVERE LOGIC ERROR OCCURRED IN PHASE=nnnnnnnn, RC=xxxx |E|IPW$$CLD |
(MI)SP -PG |SF      |RED  | | - |$GAM D=L |                                                                 |E|IPW$$CV  |
(MI)SP -PG |SF      |RED  | | - |         |                                                                 |E|IPW$$DM  |
(MI)SP -PG |SF      |RED  | | - |$GAM D=L |                                                                 |E|IPW$$DP  |
(MI)SP -PG |SF      |RED  | | - |         |                                                                 |E|IPW$$LO  |
(MI)SP -PG |SF      |RED  | | - |         |                                                                 |E|IPW$$LU  |
(MI)SP -PG |SF      |RED  | | - |$GAM D=L |                                                                 |E|IPW$$LW  |
(MI)SP -PG |SF      |RED  | | - |         |                                                                 |E|IPW$$NU  |
(MI)SP -PG |SF      |RED  | | - |         |                                                                 |E|IPW$$PC  |
(MI)SP -PG |SF      |RED  | | - |$GAM D=L |                                                                 |E|IPW$$PS  |
(MI)SP -PG |SF      |RED  | | - |         |                                                                 |E|IPW$$QM  |
(MI)SP -PG |SF      |RED  | | - |         |                                                                 |E|IPW$$RY  |
(MI)SP -PG |SF      |RED  | | - |         |                                                                 |E|IPW$$SC  |
(MI)SP -PG |SF      |RED  | | - |$GAM D=L |                                                                 |E|IPW$$T1  |
```

Figure 136 (Page 13 of 22). Message Reference

Routing Code RT= xx xx xx	Descrip- tor Code DC= xx xx xx	Color	C m n d	DOM 'ed	Issued via:	Message: (E) Module has IPW$GMM msg EQUATE $1xxxx ----------------+ (L) Module has locally defined message +--- Msg equate suffix $1xxx(n) if multiple messages v		Module: (where issued)
(MI)SP -PG	SF	RED		-	$GAM D=L		E	IPW$$TR
(MI)SP -PG	SF	RED		-			E	IPW$$TQ
(MI)SP -PG	SF	RED		-	$GAM D=L		E	IPW$$XJ
(MI)SP -PG	SF	RED		-	$GAM D=L		E	IPW$$XRE
(MA)	(DA)(CM)	WHITE	x	-	$GAM+$WTR	1QZ1D SUBSYSTEM RUNNING IN PARTITION xx - REPLY 'YES' TO ...	E	IPW$$CF
MA	DA	WHITE		-		1QZ2A (prefix of messages received from PSF or CICS Spooler)	L	IPW$$MS
(MI)		green		-		1QZ2I (prefix of messages received from PSF or CICS Spooler)	L	IPW$$MS
(MI)		green		-	$GAM+$WTR	1QZ3D PROCESS '...cmd...'? CONFIRM WITH 'YES', ELSE 'NO'	E	IPW$$CP
(MI)		green		-	$GAM+$WTR		E	IPW$$CM
(MI)		green		-		1R02I LINE cuu STOPPED, TIME=......	E	IPW$$LM
(MI)		green		-		1R03I TRANSM xxxxx, TIMEOUTS xxxxx, ERRORS xxxxx	E	IPW$$LD3
(MI)		green		-			E	IPW$$LM
(MI)		green		-		1R04I LINE cuu FORCED TO STOP BY PSTOP FORCE COMMAND, TIME=.	E	IPW$$LM
(MI)		green		-		1R05I SENT xxxxx RECEIVED xxxxx	E	IPW$$LD3
(or)	(CM)	green	x	-		1R06I LINE cuu NOT TRANSPARENT	E	IPW$$CPS
(MI)	II	green		-		1R07I TIMEOUT LIMIT IS EXCEEDED FOR switch|leased LINE cuu ..	E	IPW$$LM
(MI)		green		-		1R08I LINE cuu WAITING FOR SIGNON, TIME=.......	E	IPW$$LM
(MI)	II	green		-		1R09I LINE ERROR OCCURRED ON LINE cuu	E	IPW$$LM
(MI)		green		-		1R10I INVALID SETUP COMMAND	E	IPW$$BR
(MI)		green		-		1R11I INVALID STOP COMMAND	E	IPW$$BR
(MI)		green		-		1R12I(0)INVALID CLASS SPECIFICATION	E	IPW$$BR
(MI)		green		-		(1)INVALID OPTION SPECIFICATION	E	IPW$$BR
				-		1R13I INVALID TASK SPECIFICATION (R)	E	IPW$$BR
				-		1R14I EOF ON THE READER (R)	E	IPW$$BR
(MI)		green		-		1R15I REMOTE remid SIGNED-ON ON LINE cuu, TIME=............	E	IPW$$BR
(MI)		green		-		1R16I REMOTE remid SIGNED OFF, TIME=........	E	IPW$$LM
(MI)		green		-		1R17I LINE cuu IS IN SHUTDOWN, TIME=.........	E	IPW$$LM
(MI)		green		-		1R18I REMOTE remid FORCED TO SIGNOFF, TIME=......	E	IPW$$LM
(MI)		green		-		1R19I FIRST CARD MUST BE SIGNON CARD,READER FLUSHED	E	IPW$$BR
					LOC	1R20I nnn MESSAGES DELETED (R)	E	IPW$$MS
(MI)		green		-		1R21I SIGNON IGNORED, INVALID REMOTE-ID	E	IPW$$BR
(MI)		green		-		1R22I SIGNON IGNORED, INVALID PASSWORD	E	IPW$$BR
(MI)		green		-		1R23I REMOTE remid ALREADY SIGNED ON	E	IPW$$BR
(MI)		green		-		1R24I commandcode COMMAND OUT OF SEQUENCE	E	IPW$$BR
(MI)	II	green		-		1R25I REMOTE remid RECORD FORMAT ERROR ON LINE cuu	E	IPW$$BR
(MI)	II	green		-		1R26I FOR jobnm jobno RECORD EXCEEDS SPECIFIED LIST/PUN VALUE	E	IPW$$BW
(MI)	II	green		-		1R27I REMOTE remid COMPONENT SELECT ERROR ON cuu	E	IPW$$LM
(MI)	II	green		-		1R28I DISABLE FOR LINE cuu FAILED, POWER OFF MODEM MANUALLY	E	IPW$$LM
(MI)		green		-		1R30I(0)INVALID CCW - CCB ADDR X'aaaaaa' jobname jobnumber, ..	E	IPW$$XWE
(MI)		green		-			E	IPW$$XRE
(MI)		green		-		1R30I(1)CCB=xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx, ADDR= pid	E	IPW$$XWE
(MI)		green		-			E	IPW$$XRE

Figure 136 (Page 14 of 22). Message Reference

Routing Code RT= xx xx xx	Descrip- tor Code DC= xx xx xx	Color	C m n d	DOM 'ed	Issued via:	Message: (E) Module has IPW$GMM msg EQUATE $1xxxx ----------------+ (L) Module has locally defined message +--- Msg equate suffix $1xxx(n) if multiple messages v	 v	Module: (where issued)
(MI)		green		-		1R30I(2)CCW=xxxxxxxx xxxxxxxx, ADDR= pid	E	IPW$$XWE
(MI)		green		-			E	IPW$$XRE
(MI)SP	SF	RED		-		1R31I UNABLE TO LOG TRACE AREA, RC=nnnn	E	IPW$$NU
(MI)SP	SF	RED		-		1R32I OUTPUT EXIT INTERFACE INCORRECT, RC=nnnn, PROCESSING jobname jobnumber, TASK task-id cuu STOPPED	E	IPW$$LW
MA SP	AK	RED		-	$GAM D=L	1R33A(0)WRONG JECL FROM SPOOL|SEGMENT, JOB jobnm jobno part ...	E	IPW$$XJ
MA SP	AK	RED		-	$GAM D=L	(1)WRONG JECL FROM SPOOL|SEGMENT IGNORED FOR JOB jobnm ...	E	IPW$$XJ
(MA)	(DA)	WHITE		-	$GAM+$WTR	1R33D CORRECT FULL STATEMENT task	E	IPW$$XJ
(MI)		green		-	$GAM D=L	1R33I NO VALID CORRECTION task	E	IPW$$XJ
(or)	(CM)	green	x	-		1R34I commandcode OPERAND nn NO MEANINGFUL FOR LST OR PUN Q	E	IPW$$CA
(or)	(CM)	green	x	-			E	IPW$$CM
(or)	(CM)	green	x	-		1R35I WRUN NOT APPLICABLE FOR TAPE DISPLAY	E	IPW$$CM
(MI)		green		-	$GAM D=L	1R36I jobname jobnumber WITH INCOMPLETE OR CONFLICTING ...	E	IPW$$LR
(MI)		green		-	$GAM D=L	1R37I jobname jobnumber WITH IMPROBABLE YEAR SPECIFICATION	E	IPW$$LR
(MA) (TA)	(CM)	WHITE	x	-	$GAM+$WTR	1R40D POFFLOAD WITH 'NOJNO' SPECIFIED FOR OLDNODE - OK?	E	IPW$$CO
(MA) (TA)	(DA)(CM)	WHITE	x	-	$GAM+$WTR	1R41D SPECIFY TAPE SELECTION CRITERIA OR ENTER TO QUIT	E	IPW$$CO
(or)	(CM)	green	x	-	$GAM+$WTO	1R41I(4)TAPE STATUS REPORT CANCELED BY OPERATOR	E	IPW$$PS
(or)	(CM)	green	x	-	$GAM+$WTO	(5-8)queue QUEUE P D C	E	IPW$$PS
(or)	(CM)	green	x	-	$GAM+$WTO	(9-12)queue NOTHING TO DISPLAY	E	IPW$$PS
(or)	(CM)	green	x	-	$WTO LOC	(---------------- display line --------------------)	L	IPW$$PS
(or)	(CM)	green	x	-		1R42I commandcode OPERAND ## INCORRECT	E	IPW$$CS
(or)	(CM)	green	x	-		1R43I SHARED SPOOLING NOT ACTIVE	E	IPW$$CRE
(or)	(CM)	green	x	-		1R44I SYSID n IS OWN SYSID OR UNKNOWN	E	IPW$$CRE
(or)	(CM)	green	x	-		1R45I commandcode OPERAND ## TOO LONG	E	IPW$$CS
(or)	(CM)	green	x	-		1R46I(0)TIME IS XX/XX/XX, DATE IS XX/XX/XXXX	E	IPW$$CD
(or)	(CM)	green	x	-		(1)XXX PAGES FIXED, XXX CURRENT TASKS	E	IPW$$CD
(or)	(CM)	green	x	-		(2)NOTHING TO DISPLAY	E	IPW$$CD
(or)	(CM)	green	x	-			E	IPW$$CI
(or)	(CM)	green	x	-		(3)SYSID=n,NODEID=nodename,SECNODE=seczone	E	IPW$$CD
(or)	(CM)	green	x	-	$GAM+$WTO	(4)STATUS REPORT CANCELED BY OPERATOR	E	IPW$$PS
(or)	(CM)	green	x	-	$GAM+$WTO	(5)queue QUEUE P D C S CARDS B	E	IPW$$PS
(or)	(CM)	green	x	-	$GAM+$WTO	(6-8)queue QUEUE P D C S PAGES CC FORM B	E	IPW$$PS
(or)	(CM)	green	x	-	$GAM+$WTO	(9-12)queue NOTHING TO DISPLAY	E	IPW$$PS
(or)	(CM)	green	x	-	$GAM+$WTO	(13)queue QUEUE P D C S CARDS B (WAIT FOR RUN SUBQUEUE)	E	IPW$$PS
(or)	(CM)	green	x	-	$GAM+$WTO	(14)queue NOTHING TO DISPLAY	E	IPW$$PS
(or)	(CM)	green	x	-	$WTO LCD*	NON-LOCAL SYSID=n SECNODE=yyyyyyyy	L	IPW$$CD
(or)	(CM)	green	x	-	$WTO LOC	(----- queue entry display line -------------------)	L	IPW$$PS
(or)	(CM)	green	x	-	$WTO LOC	VSE/POWER STATUS REPORT	L	IPW$$PS
(or)	(CM)	green	x	-	$WTO LOC	(--- status report display line -------------------)	L	IPW$$PS
(or)	(CM)	green	x	-	$WTO LCD*	(--- trace information PDISPLAY TRINFO -----------)	L	IPW$$CD
						STXIT OC Exit messages (response to MSG F1,DATA=cmd):		
(or)	(CM)	green	x	-	WTO1 LOC	NO COMMAND PASSED VIA MSG INTERVACE	L	IPW$$CM
(or)	(CM)	green	x	-	WTO1 LOC	COMMAND cccccccc NOT SUPPORTED VIA MSG INTERFACE	L	IPW$$CM
(or)	(CM)	green	x	-	WTO1 LOC	(------- display line MSG F1,DATA=PDISPLAY A ------)	L	IPW$$CM
(or)	(CM)	green	x	-	WTO1 LOC	*** BEGIN OF DISPLAYING VSE/POWER TCB'S ***	L	IPW$$CM
(or)	(CM)	green	x	-	WTO1 LOC	TID ,CUU,TCBADR,T,PHASE(ADDR),REG12 ,STATE(RX)	L	IPW$$CM
(or)	(CM)	green	x	-	WTO1 LOC	(------- display line MSG F1,DATA=PDISPLAY TASKS ---)	L	IPW$$CM
(or)	(CM)	green	x	-	WTO1 LOC	*** END OF DISPLAYING VSE/POWER TCB'S ***	L	IPW$$CM
(or)	(CM)	green	x	-	WTO1 LOC	001 ** BEGIN OF DISPLAYING SPOOLED DEVICES **	L	IPW$$CM
(or)	(CM)	green	x	-	WTO1 LOC	002 PARTITION,DEV-CLASS: CUU,CUU,...	L	IPW$$CM
(or)	(CM)	green	x	-	WTO1 LOC	(------- display line MSG F1,DATA=PDISPLAY SPDEV ---)	L	IPW$$CM
(or)	(CM)	green	x	-	WTO1 LOC	nnn ** END OF DISPLAYING SPOOLED DEVICES **	L	IPW$$CM
(or)	(CM)	green	x	-	WTO1 LOC	001 ** BEGIN OF DISPLAYING SPOOLED DEVICE WITH DEV ...	L	IPW$$CM
(or)	(CM)	green	x	-	WTO1 LOC	002 PARTITION,DEV-CLASS: CUU(PUB-CODE,DEV-TYPE),CUU(..	L	IPW$$CM
(or)	(CM)	green	x	-	WTO1 LOC	(------- display line MSG F1,DATA=PDISPLAY SPDEVT --)	L	IPW$$CM
(or)	(CM)	green	x	-	WTO1 LOC	nnn ** END OF DISPLAYING SPOOLED DEVICE WITH DEV...	L	IPW$$CM
(or)	(CM)	green	x	-	WTO1 LOC	001 *** BEGIN OF AUTOSTART INFORMATION ***	L	IPW$$CM

Figure 136 (Page 15 of 22). Message Reference

```
  Routing   |Descrip-|Color|C|DOM|Issued   |Message:                                                          | Module: |
  Code      |tor Code|     |m|'ed| via:    |       (E) Module has IPW$GMM msg EQUATE  $1xxxx ----------------+|(where   |
  RT=       |DC=     |     |n|   |         |       (L) Module has locally defined message                    || issued) |
            |        |     |d|   |         |       +--- Msg equate suffix  $1xxx(n) if multiple messages     ||         |
  xx xx xx  |xx xx xx|     | |   |         |       v                                                         v|         |

 (or)       |  (CM)  |green|x| - |WTO1 LOC |        002 NO AUTOSTART STATEMENTS PROCESSED                  |L|IPW$$CM  |
 (or)       |  (CM)  |green|x| - |WTO1 LOC |        nnn REPLY TO MSG '1Q11D FORMAT QUEUES=':  (=NO AS ...  |L|IPW$$CM  |
 (or)       |  (CM)  |green|x| - |WTO1 LOC |        (------- display line MSG F1,DATA=PDISPLAY AUSTMT --)  |L|IPW$$CM  |
 (or)       |  (CM)  |green|x| - |WTO1 LOC |        nnn ***   END OF AUTOSTART INFORMATION   ***           |L|IPW$$CM  |
  ----------+--------+-----+-+---+---------+----------------------------------------------------------------+-+---------+
 (or)       |  (CM)  |green|x| - |$WTO LCD*| 1R47I   (--- task,cuu    messages pending   PDISPLAY M  ---)   |L|IPW$$CD  |
 (or)       |  (CM)  |green|x| - |         |         NO MESSAGES PENDING                                    |E|IPW$$CD  |
  ----------+--------+-----+-+---+---------+----------------------------------------------------------------+-+---------+
 (or)       |  (CM)  |green|x| - |$WTO LCD*| 1R48I   (--- task,cuu,class,num. buff,jobname,... PDISPLAY A--)|L|IPW$$CD  |
 (or)       |  (CM)  |green|x| - |         |      (0)NO READER OR WRITER TASK CURRENTLY ACTIVE              |E|IPW$$CD  |
 (or)       |  (CM)  |green|x| - |WTO1 LOC |         NO COMMAND PASSED VIA MSG INTERFACE           (OC-Exit)|L|IPW$$CM  |
 (or)       |  (CM)  |green|x| - |WTO1 LOC |         COMMAND cccccccc NOT SUPPORTED VIA MSG INTERF (OC-Exit)|L|IPW$$CM  |
  ----------+--------+-----+-+---+---------+----------------------------------------------------------------+-+---------+
 (or)       |  (CM)  |green|x| - |$GAM+$WTO| 1R49I(0)NO ACCOUNTING SUPPORT                                  |E|IPW$$CD  |
 (or)       |  (CM)  |green|x| - |$GAM+$WTO|      (1)nnnn FREE QUEUE RECORDS - QUEUE FILE nn% FULL          |E|IPW$$CD  |
 (or)       |  (CM)  |green|x| - |$GAM+$WTO|      (2)nnnnnnnn FREE DBLK GROUPS - DATA FILE nn% FULL         |E|IPW$$CD  |
 (or)       |  (CM)  |green|x| - |$GAM+$WTO|      (3)ACCOUNT FILE xx% FULL                                  |E|IPW$$CD  |
 (or)       |  (CM)  |green|x| - |$GAM+$WTO|      (4)CURRENT DBLK SIZE=nnnn, DBLK GROUP SIZE=nnnnn          |E|IPW$$CD  |
 (or)       |  (CM)  |green|x| - |$GAM+$WTO|      (5)ACCOUNT FILE EXTENT ON CKD-cuu SYS...                  |E|IPW$$CD  |
 (or)       |  (CM)  |green|x| - |$GAM+$WTO|      (6)QUEUE FILE EXTENT   ON CKD-cuu SYS...                  |E|IPW$$CD  |
 (or)       |  (CM)  |green|x| - |$GAM+$WTO|      (7)DATA FILE EXTENT n  ON CKD-cuu SYS...                  |E|IPW$$CD  |
 (or)       |  (CM)  |green|x| - |$GAM+$WTO|      (8)DATA FILE EXTENT n  ON CKD-cuu SYS...                  |E|IPW$$CD  |
 (or)       |  (CM)  |green|x| - |$GAM+$WTO|      (9)ACCOUNT FILE EXTENT ON FBA-cuu SYS...                  |E|IPW$$CD  |
 (or)       |  (CM)  |green|x| - |$GAM+$WTO|     (10)QUEUE FILE EXTENT   ON FBA-cuu SYS...                  |E|IPW$$CD  |
 (or)       |  (CM)  |green|x| - |$GAM+$WTO|     (11)DATA FILE EXTENT n  ON FBA-cuu SYS...                  |E|IPW$$CD  |
 (or)       |  (CM)  |green|x| - |$GAM+$WTO|     (12)DATA FILE EXTENT n  ON FBA-cuu SYS...                  |E|IPW$$CD  |
 (or)       |  (CM)  |green|x| - |$GAM+$WTO|     (13)USED QUEUE RECORDS ... CRE-Q ... DEL-Q ...             |E|IPW$$CD  |
 (or)       |  (CM)  |green|x| - |$GAM+$WTO|     (14RDR-Q ... LST-Q ... PUN-Q ... XMT-Q ...   .             |E|IPW$$CD  |
  ----------+--------+-----+-+---+---------+----------------------------------------------------------------+-+---------+
 (or)       |  (CM)  |green|x| - |$GAM+$WTO| 1R4AI   EXITTYPE STATE    NAME     WA-SIZE ADDRESS  EXITSIZE WU|E|IPW$$PS  |
 (or)       |  (CM)  |green|x| - |$WTO LOC |         (---------------- display line   PDISPLAY EXIT ----)   |L|IPW$$PS  |
  ----------+--------+-----+-+---+---------+----------------------------------------------------------------+-+---------+
 (or)       |  (CM)  |green|x| - |$GAM+$WTO| 1R4BI(0)DELETION QUEUE    P D C I LINES B                      |E|IPW$$PS  |
 (or)       |  (CM)  |green|x| - |$GAM+$WTO|      (1)DELETION QUEUE NOTHING TO DISPLAY                      |E|IPW$$PS  |
 (or)       |  (CM)  |green|x| - |$GAM+$WTO|      (2)CREATE   QUEUE        C I LINES B DBGP QNUM TASK OWNER |E|IPW$$PS  |
 (or)       |  (CM)  |green|x| - |$GAM+$WTO|      (3)CREATE   QUEUE NOTHING TO DISPLAY                      |E|IPW$$PS  |
 (or)       |  (CM)  |green|x| - |$WTO LOC |         (---------------- Queue Entry Display Line --------)   |L|IPW$$PS  |
 (or)       |  (CM)  |green|x| - |$WTO LOC |      (4)nn BIGGEST SORTED C I CARD/LINE  DBGP QNUM SUF   PAGES |L|IPW$$PS  |
 (or)       |  (CM)  |green|x| - |$GAM+$WTO|      (5)BIGGEST ENTRIES - NOTHING TO DISPLAY                   |E|IPW$$PS  |
  ----------+--------+-----+-+---+---------+----------------------------------------------------------------+-+---------+
 (MA)       |(DA)(CM)|WHITE|x| - |$GAM+$WTO| 1R50D(0)partition-id READER=                                   |E|IPW$$CS  |
 (MA)       |(DA)(CM)|WHITE|x| - |$GAM+$WTO|      (1)partition-id PRINTERS=                                 |E|IPW$$CS  |
 (MA)       |(DA)(CM)|WHITE|x| - |$GAM+$WTO|      (2)partition-id PUNCHES=                                  |E|IPW$$CS  |
  ----------+--------+-----+-+---+---------+----------------------------------------------------------------+-+---------+
 (or)       |  (CM)  |green|x| - |         | 1R51I(0)commandcode NO STATUS REPORT IN PROGRESS               |E|IPW$$CC  |
 (or)       |  (CM)  |green|x| - |         |      (1)commandcode OPERAND n DESIGNATES NON-EXISTING TASK     |E|IPW$$CA  |
 (or)       |  (CM)  |green|x| - |         |                                                                |E|IPW$$CP  |
 (or)       |  (CM)  |green|x| - |         |                                                                |E|IPW$$CT  |
 (or)       |  (CM)  |green|x| - |         |      (2)commandcode NON-EXISTING TASK DESIGNATED               |E|IPW$$CF  |
 (or)       |  (CM)  |green|x| - |         |                                                                |E|IPW$$CPF |
 (or)       |  (CM)  |green|x| - |         |                                                                |E|IPW$$CG  |
  ----------+--------+-----+-+---+---------+----------------------------------------------------------------+-+---------+
 (or)       |  (CM)  |green|x| - |         | 1R52I(0)commandcode LAST OPERAND INVALID                       |E|IPW$$CM  |
 (or)       |  (CM)  |green|x| - |         |      (1)commandcode INVALID DESTINATION SPECIFIED              |E|IPW$$CB  |
 (or)       |  (CM)  |green|x| - |         |      (2)commandcode OPERAND ## INVALID OR NON-EXISTING PART'N  |E|IPW$$CD  |
 (or)       |  (CM)  |green|x| - |         |      (3)commandcode OPERAND ## INVALID                         |E|IPW$$CD  |
 (or)       |  (CM)  |green|x| - |         |                                                                |E|IPW$$CE  |
 (or)       |  (CM)  |green|x| - |         |                                                                |E|IPW$$CF  |
 (or)       |  (CM)  |green|x| - |         |                                                                |E|IPW$$CG  |
 (or)       |  (CM)  |green|x| - |         |                                                                |E|IPW$$CL  |
 (or)       |  (CM)  |green|x| - |         |                                                                |E|IPW$$CLD |
 (or)       |  (CM)  |green|x| - |         |                                                                |E|IPW$$CM  |
 (or)       |  (CM)  |green|x| - |         |                                                                |E|IPW$$CP  |
 (or)       |  (CM)  |green|x| - |         |                                                                |E|IPW$$CS  |
 (or)       |  (CM)  |green|x| - |         |                                                                |E|IPW$$CT  |
 (or)       |  (CM)  |green|x| - |         |                                                                |E|IPW$$CU  |
 (or)       |  (CM)  |green|x| - |         |                                                                |E|IPW$$CV  |
 (or)       |  (CM)  |green|x| - |         |                                                                |E|IPW$$CX  |
 (or)       |  (CM)  |green|x| - |         |      (4)commandcode OPERAND ## MISSING OR INVALID              |E|IPW$$CA  |
 (or)       |  (CM)  |green|x| - |         |                                                                |E|IPW$$CAC |
 (or)       |  (CM)  |green|x| - |         |                                                                |E|IPW$$CB  |
 (or)       |  (CM)  |green|x| - |         |                                                                |E|IPW$$CC  |
 (or)       |  (CM)  |green|x| - |         |                                                                |E|IPW$$CD  |
 (or)       |  (CM)  |green|x| - |         |                                                                |E|IPW$$CG  |
```

Figure 136 (Page 16 of 22). Message Reference

Routing	Descrip-	Color	C	DOM	Issued	Message:		Module:
Code	tor Code		m	'ed	via:	(E) Module has IPW$GMM msg EQUATE $1xxxx ----------------+		(where
RT=	DC=		n			(L) Module has locally defined message		issued)
			d			+--- Msg equate suffix $1xxx(n) if multiple messages		
xx xx xx	xx xx xx					v	v	
(or)	(CM)	green	x	-			E	IPW$$CH
(or)	(CM)	green	x	-			E	IPW$$CI
(or)	(CM)	green	x	-			E	IPW$$CJ
(or)	(CM)	green	x	-			E	IPW$$CL
(or)	(CM)	green	x	-	$GAM D=L		E	IPW$$CLD
(or)	(CM)	green	x	-			E	IPW$$CN
(or) (TA)	(CM)	green	x	-			E	IPW$$CO
(or)	(CM)	green	x	-			E	IPW$$CP
(or)	(CM)	green	x	-			E	IPW$$CPF
(or)	(CM)	green	x	-			E	IPW$$CPS
(or)	(CM)	green	x	-			E	IPW$$CR
(or)	(CM)	green	x	-			E	IPW$$CRE
(or)	(CM)	green	x	-			E	IPW$$CS
(or)	(CM)	green	x	-			E	IPW$$CT
(or)	(CM)	green	x	-			E	IPW$$CU
(or)	(CM)	green	x	-			E	IPW$$CV
(or)	(CM)	green	x	-			E	IPW$$CX
(or)	(CM)	green	x	-		(5)commandcode OPERAND ## NO VALID QUEUE	E	IPW$$CD
(or)	(CM)	green	x	-			E	IPW$$CH
(or)	(CM)	green	x	-			E	IPW$$CL
(or) (TA)	(CM)	green	x	-			E	IPW$$CO
(or)	(CM)	green	x	-			E	IPW$$CR
(or)	(CM)	green	x	-		(6)commandcode INVALID SPECIFICATION FOR KEYWORD	E	IPW$$CA
(or)	(CM)	green	x	-	$GAM D=L		E	IPW$$CLD
(or)	(CM)	green	x	-			E	IPW$$CM
(or)	(CM)	green	x	-	$GAM+$WTO		E	IPW$$CD
(or)	(CM)	green	x	-		(7)commandcode OPERAND ## NOT SPECIFIED AS VALID KEYWORD	E	IPW$$CA
(or)	(CM)	green	x	-	$GAM D=L		E	IPW$$CLD
(or)	(CM)	green	x	-			E	IPW$$CM
(or)	(CM)	green	x	-		(8)commandcode NO SEARCH TYPE OPERAND SPECIFIED	E	IPW$$CA
(or)	(CM)	green	x	-		(9)commandcode INVALID BUFFER SPECIFICATION	E	IPW$$CS
(or)	(CM)	green	x	-		(10)commandcode OPERAND ## NO DEVICE ADDRESS	E	IPW$$CG
(or)	(CM)	green	x	-		(11)commandcode OPERANDS ARE INCONSISTENT	E	IPW$$CD
(or)	(CM)	green	x	-			E	IPW$$CM
(or)	(CM)	green	x	-		(12)comandcode OPERAND CPAGES AND CCARDS MUTUALLY EXCLUSIVE	E	IPW$$CM
(or)	(CM)	green	x	-		1R53I commandcode INVALID DENSITY	E	IPW$$CJ
(or) (TA)	(CM)	green	x	-			E	IPW$$CO
(or)	(CM)	green	x	-		1R54I commandcode CLASS c INVALID	E	IPW$$CM
(or)	(CM)	green	x	-		1R55I commandcode INVALID FILENAME	E	IPW$$CJ
(or)	(CM)	green	x	-		1R56I(0)lineaddr NOT INITIATED	E	IPW$$CI
(or)	(CM)	green	x	-		(1)luname PROCESSING remid	E	IPW$$CI
(or)	(CM)	green	x	-		(2)lineaddr INACTIVE	E	IPW$$CI
(or)	(CM)	green	x	-		(3)NO LOGICAL UNIT LOGGED ON	E	IPW$$CI
(or)	(CM)	green	x	-		(4)lineaddr PROCESSING remid	E	IPW$$CI
(or)	(CM)	green	x	-		(5)luname LOGGED ON	E	IPW$$CI
(or)	(CM)	green	x	-		(6)luname NOT LOGGED ON	E	IPW$$CI
(or)	(CM)	green	x	-		(7)luname LOGGING ON	E	IPW$$CI
(or)	(CM)	green	x	-		(8)cuu PROCESSING NODE nodeid	E	IPW$$CI
(or)	(CM)	green	x	-		(9)cuu NODE nodeid SESSION PENDING	E	IPW$$CI
(or)	(CM)	green	x	-		(10)NODE nodeid INACTIVE OR UNKNOWN	E	IPW$$CI
(or)	(CM)	green	x	-	$WTO LCI*	JOB-TRANSMITTER 1=A 2=I......	L	IPW$$CI
(or)	(CM)	green	x	-	$WTO LCI*	OUT-TRANSMITTER 1=A 2=I......	L	IPW$$CI
(or)	(CM)	green	x	-	$WTO LCI*	JOB-RECEIVER 1=A 2=I.....	L	IPW$$CI
(or)	(CM)	green	x	-	$WTO LCI*	OUT-RECEIVER 1=A 2=I.....	L	IPW$$CI
(or)	(CM)	green	x	-	$WTO LCI*	LOCAL NODE ACTING AS SERVER|CLIENT, CIPHER=	L	IPW$$CI
(or)	(CM)	green	x	-	$WTO LCI*	TRANSMISSION SUSPENDED, RFCS=..	L	IPW$$CI
(or)	(CM)	green	x	-	$WTO LCI*	RECEIVING SUSPENDED, TFCS=...	L	IPW$$CI
(or)	(CM)	green	x	-		1R57I(0)commandcode COMMAND IGNORED, TASK IS AT JOB BOUNDARY	E	IPW$$CT
(or)	(CM)	green	x	-			E	IPW$$CF
(MI)		green	x	-	$GAM D=L	(1)JOBEXIT FLUSH IGNORED, TASK IS AT JOB BOUNDARY	E	IPW$$LR
(MI)SP	SF	RED	x	-	$GAM D=L	(2)JOBEXIT RETURN CODE INCORRECT, TASK taskid,cuu FLUSHED	E	IPW$$LR
(or)	(CM)	green	x	-		1R58I(0)commandcode DEVICE cuu IS NOT KNOWN	E	IPW$$CJ
(or)	(CM)	green	x	-			E	IPW$$CM
(or) (TA)	(CM)	green	x	-			E	IPW$$CO
(or)	(CM)	green	x	-			E	IPW$$CS
(or)	(CM)	green	x	-		(1)commandcode DEVICE cuu IN USE	E	IPW$$CM
(or)	(CM)	green	x	-		(2)commandcode DEVICE cuu IS DOWN	E	IPW$$CM

Figure 136 (Page 17 of 22). Message Reference

```
 Routing   |Descrip-|Color|C|DOM|Issued   |Message:                                                          | Module: |
 Code      |tor Code|     |m|'ed| via:    |       (E) Module has IPW$GMM msg EQUATE  $1xxxx ----------------+|(where   |
 RT=       |DC=     |     |n|   |         |       (L) Module has locally defined message                    || issued) |
           |        |     |d|   |         |       +--- Msg equate suffix  $1xxx(n) if multiple messages     ||         |
 xx xx xx  |xx xx xx|     | |   |         |       v                                                         v|         |

 ----------+--------+-----+-+---+---------+----------------------------------------------------------------+-+---------+
(or)       |  (CM)  |green|x| - |$GAM D=L | 1R59I   FOR nodeid,userid EXECUTING COMMAND: command operand   |E|IPW$$CM  |
 ----------+--------+-----+-+---+---------+----------------------------------------------------------------+-+---------+
(or)       |  (CM)  |green|x| - |         | 1R5AI   FLUSH IGNORED, TASK IS IN STOP STATE                   |E|IPW$$CC  |
(or)       |  (CM)  |green|x| - |         |                                                                |E|IPW$$CF  |
 ----------+--------+-----+-+---+---------+----------------------------------------------------------------+-+---------+
(or)       |  (CM)  |green|x| - |         | 1R5BI   commandcode COMMAND IGNORED, RC=nnnn                   |E|IPW$$CF  |
(or)       |  (CM)  |green|x| - |         |                                                                |E|IPW$$CM  |
(or)  (TA) |  (CM)  |green|x| - |         |                                                                |E|IPW$$CO  |
(or)       |  (CM)  |green|x| - |         |                                                                |E|IPW$$CRE |
 ----------+--------+-----+-+---+---------+----------------------------------------------------------------+-+---------+
(or)       |  (CM)  |green|x| - |$GAM D=L | 1R5CI   PHASE TO BE LOADED UNSUITABLE FOR CURRENT ENVIRONMENT  |E|IPW$$CLD |
 ----------+--------+-----+-+---+---------+----------------------------------------------------------------+-+---------+
(or)       |  (CM)  |green|x| - |         | 1R5DI   commandcode COMMAND IGNORED, TRACING COULD NOT BE ...  |E|IPW$$CP  |
 ----------+--------+-----+-+---+---------+----------------------------------------------------------------+-+---------+
(MA)       |(DA)(CM)|WHITE|x| - |$GAM+$WTR| 1R60D   CONFIRM PRESET COMMAND FOR SYSID ...                   |E|IPW$$CRE |
 ----------+--------+-----+-+---+---------+----------------------------------------------------------------+-+---------+
(or)       |  (CM)  |green|x| - |         | 1R61I   commandcode INVALID FOR WRITER-ONLY PARTITION          |E|IPW$$CF  |
 ----------+--------+-----+-+---+---------+----------------------------------------------------------------+-+---------+
(or)       |  (CM)  |green|x| - |         | 1R62I   commandcode INVALID RJE PASSWORD                       |E|IPW$$CS  |
 ----------+--------+-----+-+---+---------+----------------------------------------------------------------+++---------+
(or)       |  (CM)  |green|x| - |         | 1R63I   commandcode partition-id PRIORITY TOO HIGH             |E|IPW$$CS  |
 ----------+--------+-----+-+---+---------+----------------------------------------------------------------+++---------+
(or)       |  (CM)  |green|x| - |         | 1R64I(0)commandcode NO FREE LUB AVAILABLE                      |E|IPW$$CM  |
(MI)       |        |green| | - |$GAM D=L |                                                                |E|IPW$$OT  |
(or)       |  (CM)  |green|x| - |         |      (1)commandcode SYSLST LUB NOT AVAILABLE                   |E|IPW$$CM  |
(MI)       |        |green| | - |$GAM D=L |      (2)SYSLST LUB NOT AVAILABLE task,cuu                      |E|IPW$$PL  |
(MI)       |        |green| | - |$GAM D=L |      (3)NO LUB AVAILABLE, DISP FORCED TO D jobname jobnum part.|E|IPW$$OT  |
 ----------+--------+-----+-+---+---------+----------------------------------------------------------------+++---------+
(or)       |  (CM)  |green|x| - |         | 1R65I(0)commandcode RJE,SNA NOT SUPPORTED                      |E|IPW$$CP  |
(or)       |  (CM)  |green|x| - |         |      (1)commandcode RJE,SNA ALREADY STARTED                    |E|IPW$$CS  |
(or)       |  (CM)  |green|x| - |         |      (2)commandcode RJE,BSC NOT SUPPORTED                      |E|IPW$$CS  |
(or)       |  (CM)  |green|x| - |         |      (3)commandcode RJE NOT SUPPORTED                          |E|IPW$$CB  |
(or)       |  (CM)  |green|x| - |         |                                                                |E|IPW$$CD  |
(or)       |  (CM)  |green|x| - |         |      (4)RJE OR PNET NOT SUPPORTED                              |E|IPW$$CI  |
(or)       |  (CM)  |green|x| - |         |      (5)commandcode RJE,SNA NOT STARTED                        |E|IPW$$CP  |
(or)       |  (CM)  |green|x| - |$GAM D=L |      (6)commandcode DYNAMIC PARTITION SCHEDULING NOT SUPPORTED |E|IPW$$CD  |
(or)       |  (CM)  |green|x| - |         |                                                                |E|IPW$$CLD |
(or)       |  (CM)  |green|x| - |         |                                                                |E|IPW$$CV  |
 ----------+--------+-----+-+---+---------+----------------------------------------------------------------+++---------+
(or)       |  (CM)  |green|x| - |         | 1R66I(0)commandcode cuu LIST WRITER TASK DOES NOT EXIST        |E|IPW$$CU  |
(or)       |  (CM)  |green|x| - |         |      (1)commandcode NO WRITER TASK SPECIFIED                   |E|IPW$$CT  |
 ----------+--------+-----+-+---+---------+----------------------------------------------------------------+++---------+
(or)       |  (CM)  |green|x| - |         | 1R67I   commandcode OPERAND ## REDUCED TO 99                   |E|IPW$$CU  |
 ----------+--------+-----+-+---+---------+----------------------------------------------------------------+++---------+
(or)       |  (CM)  |green|x| - |         | 1R68I(0)commandcode partition-id PARTITION NOT AVAILABLE       |E|IPW$$CS  |
(or)       |  (CM)  |green|x| - |         |      (1)commandcode partition-id IS VSE/POWER PARTITION        |E|IPW$$CS  |
 ----------+--------+-----+-+---+---------+----------------------------------------------------------------+++---------+
(or)       |  (CM)  |green|x| - |         | 1R69I(0)commandcode NO ACCOUNTING SUPPORT                      |E|IPW$$CJ  |
(or)       |  (CM)  |green|x| - |         |      (1)commandcode COMMAND REJECTED,SAVE ACCOUNT ALREADY ACTIV|E|IPW$$CJ  |
 ----------+--------+-----+-+---+---------+----------------------------------------------------------------+++---------+
(or)       |  (CM)  |green|x| - |         | 1R70I   commandcode NO DEVICE ADDRESS SPECIFIED                |E|IPW$$CT  |
 ----------+--------+-----+-+---+---------+----------------------------------------------------------------+++---------+
(or)       |  (CM)  |green|x| - |         | 1R71I   commandcode OPERAND ## IS NOT A VALID device-type      |E|IPW$$CS  |
 ----------+--------+-----+-+---+---------+----------------------------------------------------------------+++---------+
(or)       |  (CM)  |green|x| - |         | 1R72I   commandcode VIRTUAL STORAGE FOR part SMALLER THAN 128K |E|IPW$$CS  |
 ----------+--------+-----+-+---+---------+----------------------------------------------------------------+++---------+
(or)       |  (CM)  |green|x| - |         | 1R73I   commandcode INVALID DEVICE TYPE FOR task               |E|IPW$$CM  |
 ----------+--------+-----+-+---+---------+----------------------------------------------------------------+++---------+
(or)       |  (CM)  |green|x| - |         | 1R74I(0)commandcode NO PRINTER ADDRESS SPECIFIED               |E|IPW$$CU  |
(or)       |  (CM)  |green|x| - |         |      (1)commandcode INVALID LINE ADDRESS                       |E|IPW$$CS  |
(or)       |  (CM)  |green|x| - |         |                                                                |E|IPW$$CPS |
(or)       |  (CM)  |green|x| - |         |      (2)commandcode INVALID DEVICE SPECIFICATION               |E|IPW$$CJ  |
(or)  (TA) |  (CM)  |green|x| - |         |                                                                |E|IPW$$CO  |
(or)       |  (CM)  |green|x| - |         |                                                                |E|IPW$$CS  |
 ----------+--------+-----+-+---+---------+----------------------------------------------------------------+-+---------+
(or)       |  (CM)  |green|x| - |         | 1R75I   partition-id AUTOSTARTED                               |E|IPW$$CS  |
 ----------+--------+-----+-+---+---------+----------------------------------------------------------------+-+---------+
(or)       |  (CM)  |green|x| - |         | 1R76I   commandcode NUMBER OF PAGES NOT DECIMAL                |E|IPW$$CU  |
 ----------+--------+-----+-+---+---------+----------------------------------------------------------------+-+---------+
(or)       |  (CM)  |green|x| - |         | 1R77I   commandcode TASK NOT WAITING FOR OPERATOR              |E|IPW$$CG  |
(or)       |  (CM)  |green|x| - |         |                                                                |E|IPW$$CU  |
 ----------+--------+-----+-+---+---------+----------------------------------------------------------------+-+---------+
(or)       |  (CM)  |green|x| - |         | 1R78I(0)DEVICE devname CONNECTION PENDING, DDS=ddsname         |E|IPW$$CI  |
```

Figure 136 (Page 18 of 22). Message Reference

Routing Code RT= xx xx xx	Descrip- tor Code DC= xx xx xx	Color	C m n d	DOM 'ed	Issued via:	Message: (E) Module has IPW$GMM msg EQUATE $1xxxx ----------------+ (L) Module has locally defined message +--- Msg equate suffix $1xxx(n) if multiple messages v	 v	Module: (where issued)
(or)	(CM)	green	x	-		(1)DEVICE devname STARTING	E	IPW$$CI
(or)	(CM)	green	x	-		(2)DEVICE devname WAITING FOR WORK	E	IPW$$CI
(or)	(CM)	green	x	-		(3)DEVICE devname WAITING FOR OPERATOR REACTIVATION	E	IPW$$CI
(or)	(CM)	green	x	-		(4)DEVICE devname ACTIVE	E	IPW$$CI
						(5)<doesn't exist>		
(or)	(CM)	green	x	-		(6)DEVICE devname INACTIVE	E	IPW$$CI
(or)	(CM)	green	x	-		(7)DEVICE devname SETUP IN PROGRESS	E	IPW$$CI
(or)	(CM)	green	x	-	$WTO LCI*	CLASSES: xxxx,<QUEUE:y><STATUS: HALTING>	L	IPW$$CI
(or)	(CM)	green	x	-		1R79I commandcode ERRONEOUS AUTOSTART CARD(S) READ'	E	IPW$$CS
(or)	(CM)	green	x	-		1R7AI PSTART READERS|PRINTERSE|PUNCHES EXPECTED BUT NOT ...	E	IPW$$CS
(or)	(CM)	green	x	-		1R80I commandcode OPTIONAL OPERANDS OF COMMAND IGNORED	E	IPW$$CS
(or)	(CM)	green	x	-		1R81I(0)commandcode MESSAGE/OPERAND DOES NOT START WITH QUOTE	E	IPW$$CM
(or)	(CM)	green	x	-		(1)commandcode OPERAND TOO LONG OR NO CLOSING QUOTE	E	IPW$$CM
(or)	(CM)	green	x	-		(2)commandcode MESSAGE TEXT WILL BE TRUNCATED	E	IPW$$CB
(or)	(CM)	green	x	-		1R82I commandcode 'PSETUP' OR 'PRESTART' IN PROGRESS	E	IPW$$CT
(or)	(CM)	green	x	-		1R83I PINQUIRE OPERAND NEITHER 'ALL' NOR LINE ADDRESS	E	IPW$$CI
(or)	(CM)	green	x	-		1R84I commandcode DELETION NOT ALLOWED OR IMPOSSIBLE	E	IPW$$CL
(or)	(CM)	green	x	-		1R85I(0)commandcode COMMAND NOT ALLOWED FOR REMOTE OPERATOR	E	IPW$$CM
(or)	(CM)	green	x	-		(1)commandcode COMMAND NOT ALLOWED FOR X-PART / USER CONS	E	IPW$$CM
(or)	(CM)	green	x	-		1R86I PLEASE SPECIFY DEVICES TO BE SPOOLED	E	IPW$$CS
(or)	(CM)	green	x	-		1R87I commandcode TOO MANY CLASSES, FIRST n PROCESSED	E	IPW$$CM
(or)	(CM)	green	x	-		1R88I(0)OK	E	IPW$$CA
(or)	(CM)	green	x	-			E	IPW$$CAC
(or)	(CM)	green	x	-			E	IPW$$CH
(or)	(CM)	green	x	-			E	IPW$$CL
(or)	(CM)	green	x	-			E	IPW$$CN
(or)	(CM)	green	x	-			E	IPW$$CP
(or)	(CM)	green	x	-			E	IPW$$CR
(or)	(CM)	green	x	-			E	IPW$$CRE
(or)	(CM)	green	x	-			E	IPW$$CS
(or)	(CM)	green	x	-	$GAM D=L		E	IPW$$CLD
(or)	(CM)	green	x	-			E	IPW$$CV
(or)	(CM)	green	x	-	$GAM D=L		E	IPW$$PS
(or)	(CM)	green	x	-		(1)NOTHING TO HOLD	E	IPW$$CH
(or)	(CM)	green	x	-		(2)NOTHING TO RELEASE	E	IPW$$CR
(or)	(CM)	green	x	-		(3)NOTHING TO DELETE	E	IPW$$CL
(or)	(CM)	green	x	-		(4)NOTHING TO ALTER	E	IPW$$CA
(or)	(CM)	green	x	-		(5)JOB jobname jobnumber CANNOT BE ALTERED	E	IPW$$CA
(or)	(CM)	green	x	-		(6)NOTHING TO CANCEL	E	IPW$$CC
(or)	(CM)	green	x	-		(7)NOTHING TO SEGMENT	E	IPW$$CSG
(or)	(CM)	green	x	-		(8)NOTHING TO COPY	E	IPW$$CY
(or)	(CM)	green	x	-		(9)JOB jobname jobno CANNOT BE COPIED	E	IPW$$CY
(or)	(CM)	green	x	-		(9)JOB jobname jobno CANNOT BE COPIED	E	IPW$$CY
(or)	(CM)	green	x	-	$GAM+$WTO	(10)OK : NNNNN ENTRY PROCESSED BY cccccccc	E	IPW$$CA
(or)	(CM)	green	x	-	$GAM+$WTO		E	IPW$$CH
(or)	(CM)	green	x	-	$GAM+$WTO		E	IPW$$CL
(or)	(CM)	green	x	-	$GAM+$WTO		E	IPW$$CR
(or)	(CM)	green	x	-	$GAM+$WTO	(11)OK : NNNNN ENTRIES PROCESSED BY cccccccc	E	IPW$$CA
(or)	(CM)	green	x	-	$GAM+$WTO		E	IPW$$CH
(or)	(CM)	green	x	-	$GAM+$WTO		E	IPW$$CL
(or)	(CM)	green	x	-	$GAM+$WTO		E	IPW$$CR
(or)	(CM)	green	x	-	$GAM+$WTO	(12)OK : WORK AREA SHOULD BE VERIFIED IN ...	E	IPW$$CLD
(or)	(CM)	green	x	-		1R89I PEND VSE/POWER INITIATION NOT COMPLETE	E	IPW$$CE
(or)	(CM)	green	x	-		1R90I commandcode INVALID TASK SPECIFICATION operand	E	IPW$$CS
(or)	(CM)	green	x	-		1R91I(0)commandcode TOO MANY OPERANDS, COMMAND REJECTED	E	IPW$$CM
(or)	(CM)	green	x	-		(1)commandcode TOO MANY OPERANDS, FIRST n PROCESSED	E	IPW$$CM
(or)	(CM)	green	x	-		1R92I ALLUSER MESSAGE QUEUE IS FULL	E	IPW$$CB
(or)	(CM)	green	x	-		1R93I(0)commandcode REMOTE remid CURRENTLY NOT SIGNED ON	E	IPW$$CB

Figure 136 (Page 19 of 22). Message Reference

Routing	Descrip-	Color	C	DOM	Issued	Message:		Module:
Code	tor Code		m	'ed	via:	(E) Module has IPW$GMM msg EQUATE $1xxxx ----------------+		(where
RT=	DC=		n			(L) Module has locally defined message			issued)
			d			+--- Msg equate suffix $1xxx(n) if multiple messages			
xx xx xx	xx xx xx					v	v	
(or)	(CM)	green	x	-		(1)commandcode NO SESSION ESTABLISHED FOR lunamexx	E	IPW$$CP
(or)	(CM)	green	x	-		1R94I INVALID DEVICE DUPLICATION	E	IPW$$CS
(or)	(CM)	green	x	-		1R95I commandcode LINE cuu NOT SUPPORTED	E	IPW$$CI
(or)	(CM)	green	x	-			E	IPW$$CM
(or)	(CM)	green	x	-		1R96I commandcode INCORRECT OPERAND nn OF COMMAND IGNORRED	E	IPW$$CI
(or)	(CM)	green	x	-		1R97I commandcode COMMAND INVALID DURING SHUTDOWN	E	IPW$$CM
(or)	(CM)	green	x	-		1R98I commandcode INVALID VSE/POWER COMMAND	E	IPW$$CM
(or)	(CM)	green	x	-		1R99I(0)VSE/POWER HAS BEEN TERMINATED	E	IPW$$CE
(or)	(CM)	green	x	-		(1)VSE/POWER IS IN SHUTDOWN	E	IPW$$CE
(or)	(CM)	green	x	-		1R9AI SHORT COMMAND 'commandcode' UNSUPPORTED DUE TO ...	E	IPW$$CM
(or)	(CM)	green	x	-		1R9BI(0)commandcode SEGMENT REQUEST IGNORED FOR DISP=I	E	IPW$$CA
(or)	(CM)	green	x	-			E	IPW$$CSG
(or)	(CM)	green	x	-		(1)commandcode SEGMENT REQUEST IGNORED DUE TO EMPTY DBLKGP	E	IPW$$CA
(or)	(CM)	green	x	-			E	IPW$$CSG
(or)	(CM)	green	x	-		(2)commandcode SEGMENT REQUEST IGNORED FOR DISP=T	E	IPW$$CA
(or)	(CM)	green	x	-			E	IPW$$CSG
(MI)		green		-		1RA0I JOB/OUTPUT jobname jobnum TRANSMITTED TO nodeid...	E	IPW$$NT
(MI)SP	II	green		-		1RA1I JOB/OUTPUT jobname jobnum NODE nodeid UNKNOWN	E	IPW$$QM
(MI)		green		-	$NTY	1RA2I(0)COMMAND FOR NODE node1 IGNORED, NODE name NOT CONNECTED	E	IPW$$MX
(MI)		green		-	$NTY	(1)NODE nodeid UNKNOWN	E	IPW$$MX
						(IPW$NTY messages will have following RT=,DC=:		
						- when arriving via PNET will have default RT=0,DC=0		
						- when routing to local node, will have IPW$GMM dflt)		
(or)	(CM)	green	x	-		1RA3I commandcode VSE/POWER NETWORKING NOT SUPPORTED	E	IPW$$CAC
(or)	(CM)	green	x	-			E	IPW$$CB
(or)	(CM)	green	x	-			E	IPW$$CD
(or)	(CM)	green	x	-			E	IPW$$CF
(or)	(CM)	green	x	-			E	IPW$$CI
(or)	(CM)	green	x	-	$GAM D=L		E	IPW$$CLD
(or)	(CM)	green	x	-			E	IPW$$CN
(or)	(CM)	green	x	-			E	IPW$$CP
(or)	(CM)	green	x	-			E	IPW$$CS
(or)	(CM)	green	x	-			E	IPW$$CV
(or)	(CM)	green	x	-			E	IPW$$CX
(or)	(CM)	green	x	-		1RA4I commandcode INVALID NODEID nodeid RC=xxxx	E	IPW$$CA
(or)	(CM)	green	x	-			E	IPW$$CD
(or)	(CM)	green	x	-	$GAM D=L		E	IPW$$CLD
(or)	(CM)	green	x	-			E	IPW$$CP
(or)	(CM)	green	x	-			E	IPW$$CPS
(or)	(CM)	green	x	-	$GAM D=L	1RA5I commandocde INVALID NETWORK DEFINITION TABLE ndtname	E	IPW$$CLD
(MI)	II	green		-		1RA6I UNABLE TO ESTABLISH CONNECTION TO NODE nodeid RC=nnnn	E	IPW$$LD3
(or)	(CM)	green	x	-		1RA7I commandcode COMMAND NOT ALLOWED ON NODE nodeid	E	IPW$$CM
(MI)		green		-		1RA8I task TASK HAS BEEN DRAINED FOR NODE nodeid	E	IPW$$NR
(MI)		green		-			E	IPW$$NR2
(MI)		green		-			E	IPW$$NT
(MI)	II	green		-		1RA9I TRANSMISSION OF JOB/OUTPUT jobname jobnum ..CANCELLED.	E	IPW$$NT
(MI)		green		-		1RB0I(0)NODE nodeid SIGNED-OFF ON LINK cuu RC=nnnn, TIME=hh...	E	IPW$$LD3
(MI)		green		-		(1)NODE nodeid STOPPED, RC=nnnn, TIME=hh/mm/ss	E	IPW$$LD3
(or)	(CM)	green	x	-		1RB1I NODE UNKNOWN OR NO PATH ESTABLISHED TO NODE nodeid	E	IPW$$CAC
(or)	(CM)	green	x	-			E	IPW$$CB
(or)	(CM)	green	x	-			E	IPW$$CN
(or)	(CM)	green	x	-			E	IPW$$CPF
(or)	(CM)	green	x	-			E	IPW$$CX

Figure 136 (Page 20 of 22). Message Reference

Routing Code RT= xx xx xx	Descrip- tor Code DC= xx xx xx	Color	C m n d	DOM 'ed	Issued via:	Message: (E) Module has IPW$GMM msg EQUATE $1xxxx (L) Module has locally defined message +--- Msg equate suffix $1xxx(n) if multiple messages v		Module: (where issued)
(MI)	II	green		-		1RB2I INVALID SIGNON RECEIVED FROM NODE nodeid, RC=nnnn	E	IPW$$LD3
(MI)		green		-		1RB3I NODE nodeid SIGNED-ON ON LINK cuu, BSIZE=nnn, TIME=hh.	E	IPW$$LD3
(or)	(CM)	green	x	-	$GAM D=L	1RB4I commandcode NETWORK DEFINITION TABLE ndtname LOADED	E	IPW$$CLD
(MI)		green		-		1RB5I job/output jobname jobnum RECEIVED FROM nodeid FOR ...	E	IPW$$NR
(MI)		green		-			E	IPW$$NR2
(MI)	II	green		-		1RB6I(0)job/output jobname jobnum FROM nodeid CANCELED, RC=nnnn	+	IPW$$NR
(MI)	II	green		-			E	IPW$$NR2
(MI)	II	green		-		(1)CONSOLE DATA FROM nodeid CANCELED, RC=nnnn	E	IPW$$NR
(MI)	II	green		-			E	IPW$$NR2
(MI)	II	green		-		(2)RECEIVER FOR nodeid CANCELED, RC=nnnn	E	IPW$$NR
(MI)SP	SF	RED		-			E	IPW$$NR2
(or)	(CM)	green	x	-	$GAM+$WTO	1RB7I(0)NODE ROUTE1 ROUTE2 AUTH BSIZ APPLID APPLID/IPADDR	E	IPW$$PS
(or)	(CM)	green	x	-	$GAM+$WTO	(1)commandcode NOTHING TO DISPLAY	E	IPW$$PS
(or)	(CM)	green	x	-	$GAM+$WTO	(2)***** NDT NAME = xxxxxxxx *****	E	IPW$$PS
(or)	(CM)	green	x	-	$WTO LOC	(---------- NDT display line --------------------)	L	IPW$$PS
(MI)		green		-	$GAM D=L	1RB8I(0)NODE nodeid HAS RESTARTED	E	IPW$$LD1
(MI)		green		-		(0)AUTOMATIC RESTART OF CONNECTION TO NODE nodeid IN PROGR	+	IPW$$LD3
(MI)SP	SF	RED		-		1RB9I NODE ATTACHED TABLE FULL OR CONTAINS ERROR ENTRIES ...	E	IPW$$LD3
(MI)SP	SF	RED		-			E	IPW$$LD5
(MI)SP	SF	RED		-	$GAM D=L		E	IPW$$TI
MA SP(TP)	AK	RED		-	$GAM D=L	1QBAA UNACCEPTABLE PARALLEL SESSION REQUEST OCCURRED FOR ...	E	IPW$$S2
(MI)	II	green		-	$GAM D=L	1RC0I BUFFER(S) LOST ON LINK WITH NODE nodeid, RC=nnnn	E	IPW$$LD1
(MI)	II	green		-	$GAM D=L		E	IPW$$LD2
(MI)		green		-	$GAM D=L	1RC1I NETWORK PROTOCOL ERROR FOR NODE nodeid, RC=nnnn	E	IPW$$LD1
(MI)		green		-	$GAM D=L		E	IPW$$LD2
(MI)		green		-			E	IPW$$LD3
(MI)SP	SF	RED		-			E	IPW$$S2
(or)	(CM)	green	x	-		1RC2I commandcode TRANSMITTER CANNOT BE ACTIVATED	E	IPW$$CAC
(or)	(CM)	green	x	-		1RC3I commandcode COMMAND REJECTED, NODE nodeid IN SHUTDOWN	E	IPW$$CAC
(MI)	II	green		-		1RC4I UNABLE TO SHUTDOWN SESSION WITH NODE nodeid	E	IPW$$S3
(MI)		green		-	$GAM D=L	1RC6I CONNECTION PENDING FOR NODE nodeid, TIME=hh/mm/ss	E	IPW$$LD1
(MI)		green		-			E	IPW$$LD3
(MI)	II	green		-		CONNECTION PENDING FOR NODE nodeid, TIME=hh/mm/ss, RC=.	E	IPW$$S2
(MI)		green		-		1RC7I NODE nodeid AWAITING CONNECTION, TIME=......	E	IPW$$S2
(or)	(CM)	green	x	-		1RC8I PSTART COMMAND IGNORED, INVALID CTCA SPECIFIED	E	IPW$$CPS
(or)	(CM)	green	x	-		1RD0I PSTART COMMAND IGNORED, VTAM TERMINATING	E	IPW$$CPS
(or)	(CM)	green	x	-		1RD1I commandcode NODE nodeid ALREADY STARTED	E	IPW$$CPS
>VSE EXCP Default <		green		-	EXCP REQ*	1RD2I VTAM OPEN FAILED, RC=nnnn	E	IPW$$S1
>VSE EXCP Default <		green		-	EXCP REQ*	1RD3I VTAM SETLOGON FAILED, RC/FDB2=nn,nn	E	IPW$$S1
>VSE EXCP Default <		green		-	EXCP REQ*	1RD4I VTAM SETLOGON QUIESCE FAILED, RC/FDB2=nn,nn	E	IPW$$S1
>VSE EXCP Default <		green		-	EXCP REQ*	1RD5I VTAM CLOSE FAILED, RC=nnnn	E	IPW$$S1
(MI)	II	green		-	$GAM D=L	1RD6I APPLID applid NOT DEFINED IN NETWORK DEFINITION TBL	E	IPW$$S2
>VSE EXCP Default <		green		-	EXCP REQ*	1RD7I LOSTERM EXIT SCHEDULED FOR NODE nodeid, REASON LOST.	E	IPW$$SE
>VSE EXCP Default <		green		-	EXCP REQ*	1RD8I VTAM macroname FAILED, RC/FDB2=nn,nn	E	IPW$$SE
(MI)	II	green		-	$GAM D=L		E	IPW$$S2
(MI)	II	green		-	$GAM D=L		E	IPW$$S3
(MI)	II	green		-	$GAM D=L		E	IPW$$LD2
(MI)	II	green		-	$GAM D=L	1RE0I VTAM NOT STARTED OR INACTIVE	E	IPW$$LD4

Figure 136 (Page 21 of 22). Message Reference

```
 Routing    |Descrip-|Color|C|DOM|Issued    |Message:                                                          | Module: |
 Code       |tor Code|     |m|'ed| via:     |       (E) Module has IPW$GMM msg EQUATE  $1xxxx ----------------+|(where   |
 RT=        |DC=     |     |n|   |          |       (L) Module has locally defined message                    || issued) |
            |        |     |d|   |          |       +--- Msg equate suffix  $1xxx(n) if multiple messages     ||         |
 xx xx xx   |xx xx xx|     | |   |          |       v                                                         v|         |

 ----------+--------+-----+-+---+----------+------------------------------------------------------------------+-+---------+
>VSE EXCP Default < |green| | - |EXCP REQ*| 1RE1I   VTAM INTERFACE CLOSED FOR NETWORKING                     |E|IPW$$S1  |
 ----------+--------+-----+-+---+----------+------------------------------------------------------------------+-+---------+
>VSE EXCP Default < |green| | - |EXCP REQ*| 1RE2I   SESSION REQUEST FROM NODE nodeid REJECTED, RC=nnnn       |E|IPW$$SE  |
(MI)       |      II|green| | - |$GAM D=L |                                                                  |E|IPW$$S2  |
 ----------+--------+-----+-+---+----------+------------------------------------------------------------------+-+---------+
(or)       |  (CM)  |green|x| - |$GAM+$WTO| 1RE3I   APPLID FOR NODE nodeid ALREADY DEFINED IN NDT            |E|IPW$$CLD |
 ----------+--------+-----+-+---+----------+------------------------------------------------------------------+-+---------+
(or)       |  (CM)  |green|x| - |$GAM+$WTO| 1RE4I   IP-ADDRESS WITH PORT FOR NODE nodeid ALREADY DEFINED ..|E|IPW$$CLD |
 ----------+--------+-----+-+---+----------+------------------------------------------------------------------+-+---------+
(or)       |  (CM)  |green|x| - |$GAM+$WTO| 1RE5I   NETWORK DEFINITION TABLE ... FOUND WITH NEW LOCAL NODE.|E|IPW$$CLD |
 ----------+--------+-----+-+---+----------+------------------------------------------------------------------+-+---------+
(or)       |  (CM)  |green|x| - |$GAM+$WTR| 1RE6D   CONFIRM CHANGE OF LOCAL NODE NAME FROM ... TO ...        |E|IPW$$CLD |
 ----------+--------+-----+-+---+----------+------------------------------------------------------------------+-+---------+
(or)       |  (CM)  |green|x| - |$GAM D=L | 1RE7I   CHANGE OF LCOAL NODE NAME FROM ... IN PROGRESS           |E|IPW$$CLD |
(or)       |  (CM)  |green|x| - |$GAM D=L | 1RE7I   CHANGE OF LCOAL NODE NAME FROM ... COMPLETED             |E|IPW$$CLD |
(or)       |  (CM)  |green|x| - |$GAM+$WTR| 1RE7I   CHANGE OF LCOAL NODE NAME FROM ...RE-INITIATED           |E|IPW$$CLD |
(or)       |  (CM)  |green|x| - |$GAM+$WTR| 1RE7I   CHANGE OF LCOAL NODE NAME FROM ...INTERRUPTED            |E|IPW$$CLD |
 ----------+--------+-----+-+---+----------+------------------------------------------------------------------+-+---------+
(or)       |  (CM)  |green|x| - |$GAM+$WTR| 1RE8I   CHANGE OF LOCAL NODE NAME FAILED, ACTIVE SYSIDS  FOUND |E|IPW$$CLD |
 ----------+--------+-----+-+---+----------+------------------------------------------------------------------+-+---------+
(or)       |  (CM)  |green|x| - |$GAM D=L | 1RE9I   queue jobname jobno jobsuffix KEPT WITH HOLD .. RC=..  |E|IPW$$I7  |
 ----------+--------+-----+-+---+----------+------------------------------------------------------------------+-+---------+
(or)       |  (CM)  |green|x| - |$GAM D=L | 1REAI   CHANGE OF LOCALNODE NAME FROM.. TO .. INCOMPLETE ...   |E|IPW$$I7  |
 ----------+--------+-----+-+---+----------+------------------------------------------------------------------+-+---------+
(or)       |  (CM)  |green|x| - |EXCP $$I7| 1REBI   LAST QUEUE ENTRY PROCESSED SUCCESSFULLY BY LOCAL ...   |L|IPW$$AT  |
 ----------+--------+-----+-+---+----------+------------------------------------------------------------------+-+---------+
(or)       |  (CM)  |green|x| - |$GAM D=L | 1RECI   STATUS REPORT $LSTNNNN BEING CREATED DUE .. NODE NAME..|E|IPW$$I7  |
 ----------+--------+-----+-+---+----------+------------------------------------------------------------------+-+---------+
(or)  (TA) |   (CM) |green|x| - |$GAM D=L |  1RF0I  commandcode OPERAND nn CURRENT DBLK SIZE mmmmm TOO BIG.|E|IPW$$CO  |
 ----------+--------+-----+-+---+----------+------------------------------------------------------------------+-+---------+
(MI)SP     |SF      |RED  | | - |$GAM D=L |  1RF1A  task,cuu EITHER ENCOUNTERED A PROBLEM WITH P390 OR ... |E|IPW$$PL  |
(MI)SP     |SF      |RED  | | - |$GAM D=L |                                                                  |E|IPW$$PP  |
 ----------+--------+-----+-+---+----------+------------------------------------------------------------------+-+---------+
(MI) SP    |SF      |RED  | | - |$GAM D=L | 1RF2A  SWITCH DEBUG ON TO SUPPORT TASK TRACE WITH OPTION 'FULL'|E|IPW$$CS  |
 ----------+--------+-----+-+---+----------+------------------------------------------------------------------+-+---------+
(or)       |  (CM)  |green|x| - |$GAM+$WTO| 1RT1I   UNABLE TO ATTACH TCP/IP SUBTASK, RC=nnnn                 |E|IPW$$CLD |
(or)       |  (CM)  |green|x| - |$GAM+$WTO|                                                                  |E|IPW$$CS  |
 ----------+--------+-----+-+---+----------+------------------------------------------------------------------+-+---------+
(MI)SP     |      II|green| | - |$GTO D=L | 1RT2I   TCP/IP: EZASMI MACRO-REQUEST req-type FAILED RC= ...   |E|IPW$$TD  |
 ----------+--------+-----+-+---+----------+------------------------------------------------------------------+-+---------+
(MI)SP     |      II|green| | - |$GTO     | 1RT3I   TCP/IP: CONNECT REQUEST RECEIVED FROM UNKNOWN NODE ... |E|IPW$$TD  |
 ----------+--------+-----+-+---+----------+------------------------------------------------------------------+-+---------+
 MA        |   DA   |WHITE| |YES|$GTO     | 1RT4A   TCP/IP: NO OPEN CONTROL RECORD RECEIVED FROM nodeed      |E|IPW$$TD  |
 ----------+--------+-----+-+---+----------+------------------------------------------------------------------+-+---------+
 MA        |   DA   |WHITE| |YES|$GTO     | 1RT5A   TCP/IP: OPEN|ACK|NAK CONTROL RECORD RECEIVED FROM node |E|IPW$$TD  |
 ----------+--------+-----+-+---+----------+------------------------------------------------------------------+-+---------+
(MI)SP     |      II|green| | - |$GTO     | 1RT6I   TCP/IP: NAK CONTROL RECORD RECEIVED FROM NODE nodeid     |E|IPW$$TD  |
 ----------+--------+-----+-+---+----------+------------------------------------------------------------------+-+---------+
(MI)SP     |      II|green| | - |$GTO     | 1RT7I   TCP/IP: INTERFACE STARTING, SOCKET CALL socketcall ... |E|IPW$$TD  |
 ----------+--------+-----+-+---+----------+------------------------------------------------------------------+-+---------+
 MA        |   DA   |WHITE| |YES|$GTO     | 1RT8A   TCP/IP: INTERFACE NOT AVAILABLE                          |E|IPW$$TD  |
 ----------+--------+-----+-+---+----------+------------------------------------------------------------------+-+---------+
(MI)SP     |      II|green| | - |$GTO     | 1RT9I   TCP/IP: INTERFACE NOT STARTED AT ALL                     |E|IPW$$TD  |
 ----------+--------+-----+-+---+----------+------------------------------------------------------------------+-+---------+
(MI)SP     |      II|green| | - |$GTO     | 1RTAI   TCP/IP: INTERFACE NOTIFIED FOR TERMINATION, RC=nnnn      |E|IPW$$TD  |
 ----------+--------+-----+-+---+----------+------------------------------------------------------------------+-+---------+
(MI)SP     |      II|green| | - |$GTO     | 1RTBI   TCP/IP: ERROR FOR HOSTNAME ip-name                       |E|IPW$$TD  |
 ----------+--------+-----+-+---+----------+------------------------------------------------------------------+-+---------+
 MA        |   DA   |WHITE| |YES|$GTO     | 1RTCA   TCP/IP: NODE nodeid WITH UNKNOWN HOSTNAME ip-name        |E|IPW$$TD  |
 ----------+--------+-----+-+---+----------+------------------------------------------------------------------+-+---------+
(MI)SP     |      II|green| | - |$GTO     | 1RTDI   TCP/IP: NO ACK/NAC CONTROL RECORD RECEIVED FROM ...      |E|IPW$$TD  |
 ----------+--------+-----+-+---+----------+------------------------------------------------------------------+-+---------+
(MI)SP     |      II|green| | - |$GTO     | 1RTEI   TCP/IP: CONNECTION CLOSED FOR UNKNOWN IP-ADDRESS=ip-add|E|IPW$$TD  |
 ----------+--------+-----+-+---+----------+------------------------------------------------------------------+-+---------+
 MA        |   DA   |WHITE| |YES|$GTO     | 1RTFA   TCP/IP: DATA FROM NODE|IP-ADDRESS = ...                  |E|IPW$$TD  |
 ----------+--------+-----+-+---+----------+------------------------------------------------------------------+-+---------+
 MA        |   DA   |WHITE| |YES|$GTO     | 1RTGA   TCP/IP: NO OPEN CONTROL RECORD RECEIVED IN TIME FROM ..|E|IPW$$TD  |
 ----------+--------+-----+-+---+----------+------------------------------------------------------------------+-+---------+
(MI)SP     |      II|green| | - |$GTO     | 1RTHI   TCP/IP: NODE nodeid AWAITING CONNECTION                  |E|IPW$$TD  |
 ----------+--------+-----+-+---+----------+------------------------------------------------------------------+-+---------+
 MA        |   DA   |WHITE| |YES|$GTO     | 1RTJA   TCP/IP: INITIALIZATION OF INTERFACE UNSUCCESSFUL, ...  |E|IPW$$TD  |
 ----------+--------+-----+-+---+----------+------------------------------------------------------------------+-+---------+
(MI)SP     |      II|green| | - |$GTO     | 1RTK1   TCP/IP: INTERNAL ERROR FOR NODE nodid, CCW=data          |E|IPW$$TD  |
```

Figure 136 (Page 22 of 22). Message Reference

```
  Routing   |Descrip-|Color|C|DOM|Issued   |Message:                                                         | Module: |
  Code      |tor Code|     |m|'ed| via:    |       (E) Module has IPW$GMM msg EQUATE  $1xxxx ---------------+|(where   |
  RT=       |DC=     |     |n|   |         |       (L) Module has locally defined message                   || issued) |
            |        |     |d|   |         |       +--- Msg equate suffix  $1xxx(n) if multiple messages    ||         |
  xx xx xx  |xx xx xx|     | |   |         |       v                                                        v|         |

  ----------+--------+-----+-+---+---------+----------------------------------------------------------------+-+---------+
 (MI)SP     |      II|green| | - |$GTO     | 1RTL1   TCP/IP: INTERNAL POSTING FOR NODE nodeid FAILED        |E|IPW$$TD  |
  ----------+--------+-----+-+---+---------+----------------------------------------------------------------+-+---------+
 (or)       |  (CM)  |green|x| - |         | 1RTMI(0)TCP/IP: SUBTASK ATTACHED                               |E|IPW$$CS  |
 (MI)SP     |      II|green| | - |$GTO     |      (1)TCP/IP: SUBTASK ALREADY STARTED                        |E|IPW$$TD  |
  ----------+--------+-----+-+---+---------+----------------------------------------------------------------+-+---------+
 (MI)SP     |      II|green| | - |$GTO     | 1RTN1   TCP/IP: CONNECTION CLOSED FOR NODE nodeid DUE TO STOP..|E|IPW$$TD  |
  ----------+--------+-----+-+---+---------+----------------------------------------------------------------+-+---------+
 (MI)SP     |      II|green| | - |$GTO     | 1RTO1   TCP/IP: CONNECTION ATTEMPT REJECTED BY NODE nodeid ....|E|IPW$$TD  |
  ----------+--------+-----+-+---+---------+----------------------------------------------------------------+-+---------+
 (MI)SP     |      II|green| | - |$GTO     | 1RTP1   TCP/IP: CONNECTION CLOSED FOR NODE nodid DUE TO INVAL..|E|IPW$$TD  |
  ----------+--------+-----+-+---+---------+----------------------------------------------------------------+-+---------+
 (MI)SP     |      II|green| | - |$GTO     | 1RTQ1   TCP/IP: CONNECTION CLOSED FOR NODE nodid DUE TO FAILIN.|E|IPW$$TD  |
  ----------+--------+-----+-+---+---------+----------------------------------------------------------------+-+---------+
 (MI)SP     |      II|green| | - |$GTO     | 1RTR1   TCP/IP: CONNECTION CLOSED FOR NODE nodid DUE TO INTERN.|E|IPW$$TD  |
  ----------+--------+-----+-+---+---------+----------------------------------------------------------------+-+---------+
 (MI)SP     |      II|green| | - |$GTO     | 1RTS1   TCP/IP: INTERFACE TO TCP/IP TERMINATED DUE TO ...      |E|IPW$$TD  |
  ----------+--------+-----+-+---+---------+----------------------------------------------------------------+-+---------+
 (MI)       |        |green| | - |WTO1 LOC | 1RTT1   (PNET console trace information)                       |E|IPW$$TD  |
 (MI)       |        |green| | - |WTO1 LOC |                                                                |E|IPW$$LD1 |
  ----------+--------+-----+-+---+---------+----------------------------------------------------------------+-+---------+
>VSE Exception Msg< |RED  | | - |EXCP LOC || 1RTUA   TCP/IP INTERFACE QUESTIONABLE DUE TO FAILURE IN TIDY...|L|IPW$$AT  |
  ----------+--------+-----+-+---+---------+----------------------------------------------------------------+-+---------+
 (MI)SP     |      II|green| | - |$GTO     | 1RTV1   TCP/IP: NEW CONNECTION REQUEST REJECTED FOR NODE nodeid|E|IPW$$TD  |
  ----------+--------+-----+-+---+---------+----------------------------------------------------------------+-+---------+
 (MI)SP     |      II|green| | - |$GTO     | 1RTW1   TCP/IP: CONNECTION CLOSED FOR NODE nodeid DUE TO NEW...|E|IPW$$TD  |
  ----------+--------+-----+-+---+---------+----------------------------------------------------------------+-+---------+
 (MI)SP     |      II|green| | - |$GTO     | 1RTX1   TCP/IP: DATA FROM NODE|IP-ADDRESS ...                  |E|IPW$$TD  |
  ----------+--------+-----+-+---+---------+----------------------------------------------------------------+-+---------+
 (MI)SP     |      II|green| | - |$GTO     | 1RTY1   TCP/IP: NEW CONNECTION REQUESTS FROM REMOTE NODES CAN..|E|IPW$$TD  |
  ----------+--------+-----+-+---+---------+----------------------------------------------------------------+-+---------+
 (MI)SP     |      II|green| | - |$GTO     | 1RTZ1   TCP/IP: CONNECTION CLOSED FOR NODE nodid DUE TO CLOSE..|E|IPW$$TD  |
  ----------+--------+-----+-+---+---------+----------------------------------------------------------------+-+---------+
 (or)SP     |  (CM)II|green| | - |$GAM+$WTO| 1RV11   UNABLE TO ATTACH TCP SSL SUBTASK, RC=..                |E|IPW$$CS  |
  ----------+--------+-----+-+---+---------+----------------------------------------------------------------+-+---------+
 (MI)SP     |      II|green| |   |$GTS     | 1RV21   TCP SSL: TOO MANY SOCKETS IN USE (...                  |E|IPW$$SD  |
  ----------+--------+-----+-+---+---------+----------------------------------------------------------------+-+---------+
 (MI)SP     |      II|green| |   |$GTS     | 1RV31   TCP SSL: RECEIVED CONNECT REQUEST .. NOT USING SSL ..  |E|IPW$$SD  |
  ----------+--------+-----+-+---+---------+----------------------------------------------------------------+-+---------+
 (MI)SP     |      II|green| |   |$GTO     | 1RV41   TCP/IP:  RECEIVED CONNECT REQUEST .. IS USING SSL ..   |E|IPW$$TD  |
  ----------+--------+-----+-+---+---------+----------------------------------------------------------------+-+---------+
 (MI)SP     |      II|green| |   |$GTS     | 1RV51   TCP SSL: CONNECT REQUEST REJECTED .. NOT USING SSL ..  |E|IPW$$SD  |
  ----------+--------+-----+-+---+---------+----------------------------------------------------------------+-+---------+
 (MI)SP     |      II|green| |   |$GTO     | 1RV61   TCP/IP:  CONNECT REQUEST REJECTED .. IS USING SSL ..   |E|IPW$$TD  |
  ----------+--------+-----+-+---+---------+----------------------------------------------------------------+-+---------+
 (MI)SP     |      II|green| |   |$GTS     | 1RV71   TCP...   WRONG NODE TYPE; REMONTE NODE ...             |E|IPW$$SD  |
  ----------+--------+-----+-+---+---------+----------------------------------------------------------------+-+---------+
 (or)SP     |  (CM)II|green| | - |$GAM+$WTO| 1RV91   cccccccc TCP SSL INTERFACE NOT STARTED AT ALL          |E|IPW$$CP  |
  ----------+--------+-----+-+---+---------+----------------------------------------------------------------+-+---------+
 (or)SP     |  (CM)II|green| | - |$GAM+$WTO| 1RVA1   cccccccc TCP SSL INTERFACE NOTIFIED FOR TERMINATION RC=|E|IPW$$CP  |
  ----------+--------+-----+-+---+---------+----------------------------------------------------------------+-+---------+
 (MI)SP     |      II|green| |   |$GTS     | 1RVB1   TCP SSL: CONNECTION COLOSED FOR .. DUE TO WRONG CIPHER.|E|IPW$$SD  |
  ----------+--------+-----+-+---+---------+----------------------------------------------------------------+-+---------+
 (or)SP     |  (CM)II|green| | - |$GAM+$WTO| 1RVM1(0)TCP SSL SUBTASK ATTACHED                               |+|IPW$$CS  |
 (or)SP     |  (CM)II|green| | - |$GAM+$WTO|      (1)TCP SSL SUBTASK ALREADY ATTACHED                       |+|IPW$$CS  |
  ----------+--------+-----+-+---+---------+----------------------------------------------------------------+-+---------+
 >VSE Exception Msg< |RED  | | - |EXCP LOC | 1RVUA   TCP SSL INTERFACE QUESTIONABLE DUE TO FAILURE TIDY-UP..|L|IPW$$AT  |
```

Message Reference Table (1Vxxx Messages Only)

Figure 137. Message Reference

```
 Routing    |Descrip-|Color|C|DOM|Issued   |Message:                                                      | Module: |
 Code       |tor Code|     |m|'ed| via:    |      (E) Module has IPW$GMM msg EQUATE  $1xxxx ----------------+|(where   |
 RT=        |DC=     |     |n|   |         |      (L) Module has locally defined message                   || issued) |
            |        |     |d|   |         |      (C)=Central Operator Msg(default),(R)=RJE Only,(B)=Both-+ ||         |
            |        |     | |   |         |      +--- Msg equate suffix  $1xxx(n) if multiple messages   | ||         |
 xx xx xx   |xx xx xx|     | |   |         |      v                                                      v v|         |
 MI SP      |      II|green| | - |         | 1V01I   NO SUBTASK AVAILABLE FOR RJE/SNA                  (C)|E|IPW$$SN  |
 ----------+--------+-----+-+---+---------+---------------------------------------------------------------+-+---------+
(MI)SP      |      II|green| | - |         | 1V02I   VTAM OPEN FAILURE RTNCD=nnnn                      (C)|E|IPW$$SN  |
 ----------+--------+-----+-+---+---------+---------------------------------------------------------------+-+---------+
(MI)SP      |      II|green| | - |         | 1V03I   ERROR ON rplrequest RTNCD,FDB2=nn,nn SENSE=yyyy   (C)|E|IPW$$SN  |
 ----------+--------+-----+-+---+---------+---------------------------------------------------------------+-+---------+
(MI)        |        |green| | - |         | 1V04I   RJE,SNA STARTED                                   (C)|E|IPW$$SN  |
 ----------+--------+-----+-+---+---------+---------------------------------------------------------------+-+---------+
(MI)        |        |green| | - |         | 1V05I   RJE,SNA TERMINATED                                (C)|E|IPW$$SN  |
 ----------+--------+-----+-+---+---------+---------------------------------------------------------------+-+---------+
(MI)        |      II|green| | - |$GAM D=L | 1V06I   UNABLE TO LOGON luname RC=nnnn MACRO=mname        (C)|E|IPW$$LH  |
(MI)        |      II|green| | - |$GAM D=L |                                                           (C)|E|IPW$$LN  |
>VSE EXCP Default < |green| | - |EXCP REQ |                                                           (C)|E|IPW$$VE  |
 ----------+--------+-----+-+---+---------+---------------------------------------------------------------+-+---------+
(MI)        |      II|green| | - |         | 1V07I   ERROR ON rplrequest RTNCD,FDB2=nn,nn SENSE=xxx ON ..(C)|E|IPW$$IB  |
(MI)        |      II|green| | - |         |                                                           (C)|E|IPW$$LH  |
(MI)        |      II|green| | - |         |                                                           (C)|E|IPW$$LN  |
(MI)        |      II|green| | - |         |                                                           (C)|E|IPW$$MP  |
(MI)        |      II|green| | - |         |                                                           (C)|E|IPW$$OB  |
 ----------+--------+-----+-+---+---------+---------------------------------------------------------------+-+---------+
(MI)        |      II|green| | - |         | 1V08I   luname BIND PARAMETERS INVALID                    (C)|E|IPW$$LH  |
 ----------+--------+-----+-+---+---------+---------------------------------------------------------------+-+---------+
(MI)        |        |green| | - |         | 1V09I   REMOTE remid LOGGED ON TO applid ON luname TIME=....(B)|E|IPW$$LN  |
 ----------+--------+-----+-+---+---------+---------------------------------------------------------------+-+---------+
(MI)        |        |green| | - |         | 1V10I   RJE,SNA IS IN SHUTDOWN, TIME=.......              (B)|E|IPW$$SN  |
 ----------+--------+-----+-+---+---------+---------------------------------------------------------------+-+---------+
(MI)        |        |green| | - |         | 1V11I   REMOTE remid LOGGED OFF FROM applid ON luname, TIME=(C)|E|IPW$$LF  |
 ----------+--------+-----+-+---+---------+---------------------------------------------------------------+-+---------+
            |        |     | |   |         | 1V12I   LOGOFF COMPLETED, TIME=......                     (R)|E|IPW$$LF  |
 ----------+--------+-----+-+---+---------+---------------------------------------------------------------+-+---------+
            |        |     | |   |         | 1V13I   LOGOFF FORCED, TIME=.....                         (R)|E|IPW$$LF  |
 ----------+--------+-----+-+---+---------+---------------------------------------------------------------+-+---------+
            |        |     | |   |         | 1V14I   SESSION IS IN SHUTDOWN, TIME=.......              (R)|E|IPW$$SN  |
 ----------+--------+-----+-+---+---------+---------------------------------------------------------------+-+---------+
            |        |     | |   |         | 1V15I   NO STORAGE AVAILABLE FOR task                     (R)|E|IPW$$OB  |
 ----------+--------+-----+-+---+---------+---------------------------------------------------------------+-+---------+
 MI SP      |      II|green| | - |         | 1V16I   NO STORAGE AVAILABLE FOR task FOR luname remid    (C)|E|IPW$$OB  |
 ----------+--------+-----+-+---+---------+---------------------------------------------------------------+-+---------+
            |        |     | |   |         | 1V17A   task SUSPENDED FOR FORMS MOUNT                    (R)|E|IPW$$OB  |
 ----------+--------+-----+-+---+---------+---------------------------------------------------------------+-+---------+
            |        |     | |   |         | 1V18A   REPLY WITH RESTART ON INTERVENTION REQUIRED task  (R)|E|IPW$$OB  |
 ----------+--------+-----+-+---+---------+---------------------------------------------------------------+-+---------+
            |        |     | |   |         | 1V22I   INVALID commandcode COMMAND                       (R)|E|IPW$$IB  |
 ----------+--------+-----+-+---+---------+---------------------------------------------------------------+-+---------+
            |        |     | |   |         | 1V23I   commandcode OUT OF SEQUENCE                       (R)|E|IPW$$IB  |
 ----------+--------+-----+-+---+---------+---------------------------------------------------------------+-+---------+
            |        |     | |   |         | 1V24I   task TERMINATED, REASON=nnnn FOR luname           (R)|E|IPW$$IB  |
            |        |     | |   |         |                                                           (R)|E|IPW$$OB  |
 ----------+--------+-----+-+---+---------+---------------------------------------------------------------+-+---------+
            |        |     | |   |         | 1V25I   EOJ ADDED FOR jobname jobnumber                   (R)|E|IPW$$IB  |
 ----------+--------+-----+-+---+---------+---------------------------------------------------------------+-+---------+
(MI)        |        |green| | - |         | 1V26I   INVALID REMOTE ID, PASSWORD OR LUNAME RC=yy       (C)|E|IPW$$LH  |
 ----------+--------+-----+-+---+---------+---------------------------------------------------------------+-+---------+
(MI)        |        |green| | - |         | 1V27I   REMID remid EXCEEDS SESSLIM                       (C)|E|IPW$$LH  |
 ----------+--------+-----+-+---+---------+---------------------------------------------------------------+-+---------+
            |        |     | |   |         | 1V28I   JOB jobname jobnumber GETVIS FOR COCB FAILED      (R)|E|IPW$$OC  |
 ----------+--------+-----+-+---+---------+---------------------------------------------------------------+-+---------+
            |        |     | |   |         | 1V29I   JOB jobname jobnumber GETVIS FOR COMPACTION TBL FAIL(R)|E|IPW$$OC  |
 ----------+--------+-----+-+---+---------+---------------------------------------------------------------+-+---------+
            |        |     | |   |         | 1V30I   JOB jobname jobnumber COMPACTION TABLE NOT FOUND   (R)|E|IPW$$OC  |
 ----------+--------+-----+-+---+---------+---------------------------------------------------------------+-+---------+
            |        |     | |   |         | 1V31I   JOB jobname jobnumber NO SPACE AVAIL. IN CMPT POOL (R)|E|IPW$$OC  |
 ----------+--------+-----+-+---+---------+---------------------------------------------------------------+-+---------+
            |        |     | |   |         | 1V32I   JOB jobname jobnumber INVALID COMPACTION TABLE     (R)|E|IPW$$OC  |
 ----------+--------+-----+-+---+---------+---------------------------------------------------------------+-+---------+
            |        |     | |   |         | 1V33I   REMOTE remid OUTPUT FOR NONWRITER WORKSTATION      (R)|E|IPW$$SN  |
 ----------+--------+-----+-+---+---------+---------------------------------------------------------------+-+---------+
(MI)        |      II|green| | - |         | 1V34I   display of bind parameters                         (C)|E|IPW$$LH  |
 ----------+--------+-----+-+---+---------+---------------------------------------------------------------+-+---------+
```

Message Reference Table (without Message Number)

Figure 138. Message Reference

```
  Routing    |Descrip-|Color|C|DOM|Issued   |Message:                                                       | Module: |
  Code       |tor Code|     |m|'ed| via:    |       (E) Module has IPW$GMM msg EQUATE  $1xxxx ----------------+|(where   |
  RT=        |DC=     |     |n|   |         |       (L) Module has locally defined message                   || issued) |
             |        |     |d|   |         |       +--- Msg equate suffix  $1xxx(n) if multiple messages     ||         |
  xx xx xx   |xx xx xx|     | |   |         |       v                                                        v|         |
 (MI)        |        |green| | - |$WTO LOC |        (--Invalid DEFINE statement--------------------------)  |L|IPW$$I2  |
 (MI)        |        |green| | - |$WTO LOC |        (--Invalid SET    statement--------------------------)  |L|IPW$$I2  |
  ----------+--------+-----+-+---+---------+-----------------------------------------------------------------+-+---------+
 (MI)        |        |green| | - |$WTO LOC |        (--Incorrect JECL statement--------------------------)  |L|IPW$$LR  |
 (MI)        |        |green| | - |$WTO LOC |        (--Incorrect JECL statement--------------------------)  |L|IPW$$XJ  |
  ----------+--------+-----+-+---+---------+-----------------------------------------------------------------+-+---------+
 (MI)        |        |green| | - |$WTO LOC |        (--Trace of PNET BSC/CTC  Input/Output Buffers-------)  |L|IPW$$LD1 |
 (MI)        |        |green| | - |$WTO LOC |        (--Display of invalid PNET CTC buffer contents-------)  |L|IPW$$LD1 |
 (MI)        |        |green| | - |$WTO LOC |        (--Display of invalid MLI character display buffer---)  |L|IPW$$LD1 |
  ----------+--------+-----+-+---+---------+-----------------------------------------------------------------+-+---------+
 (or)        |  (CM)  |green|x| - |$WTO LOC |        (--VIO dump (to printer/LST entry)-------------------)  |L|IPW$$PS  |
```

Chapter 5. Storage Layout and Data Areas

This chapter describes the storage layout of the VSE/POWER partition and the layout of the SVA part which is PFIXed by VSE/POWER. Additional it describes the control blocks, buffer areas, save areas and work spaces required by VSE/POWER.

Most VSE/POWER control blocks and many sections of VSE/POWER code are equipped with storage descriptors which serve to rapidly locate and identify important values within a storage dump. A storage descriptor is a 16-byte alphameric character string with line alignment. Where appropriate, storage descriptors may also be addressed by internal programming. For instance, the storage descriptors of some TCBs are modified dynamically to reflect the function that the TCB is performing at any given time. For example, a storage descriptor of

```
TCBb1RDR.030.000
```

indicates the start of a task control block for an RJE reader task on RJE line number 30 invoked by the central operator. Thus, a storage descriptor identified in a dump constitutes a debugging aid.

The Layout of the SVA Part of VSE/POWER

The SVA part of VSE/POWER holds the Control Address Table (CAT), the VSE/POWER Nucleus with 3 control blocks and the VSE/POWER partition control blocks for 11 (maximum) static partitions. The SVA part of VSE/POWER is shown in Figure 139.

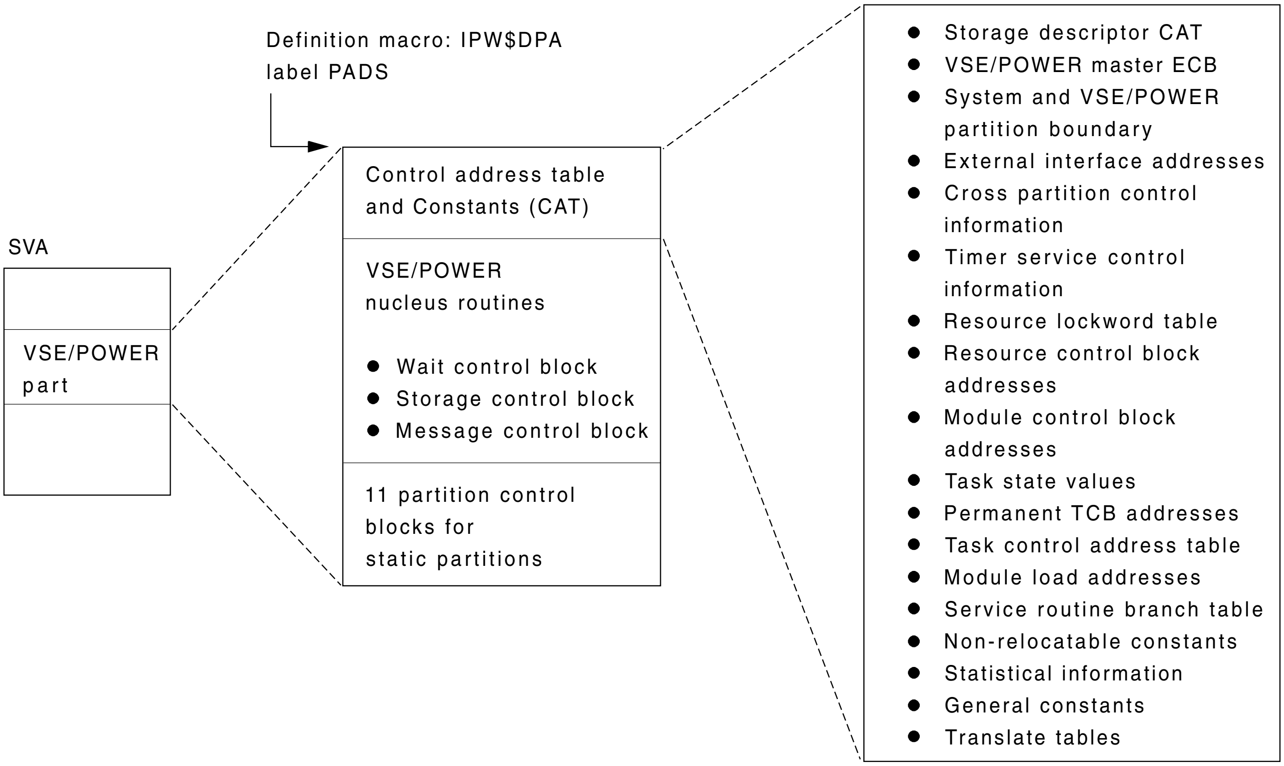


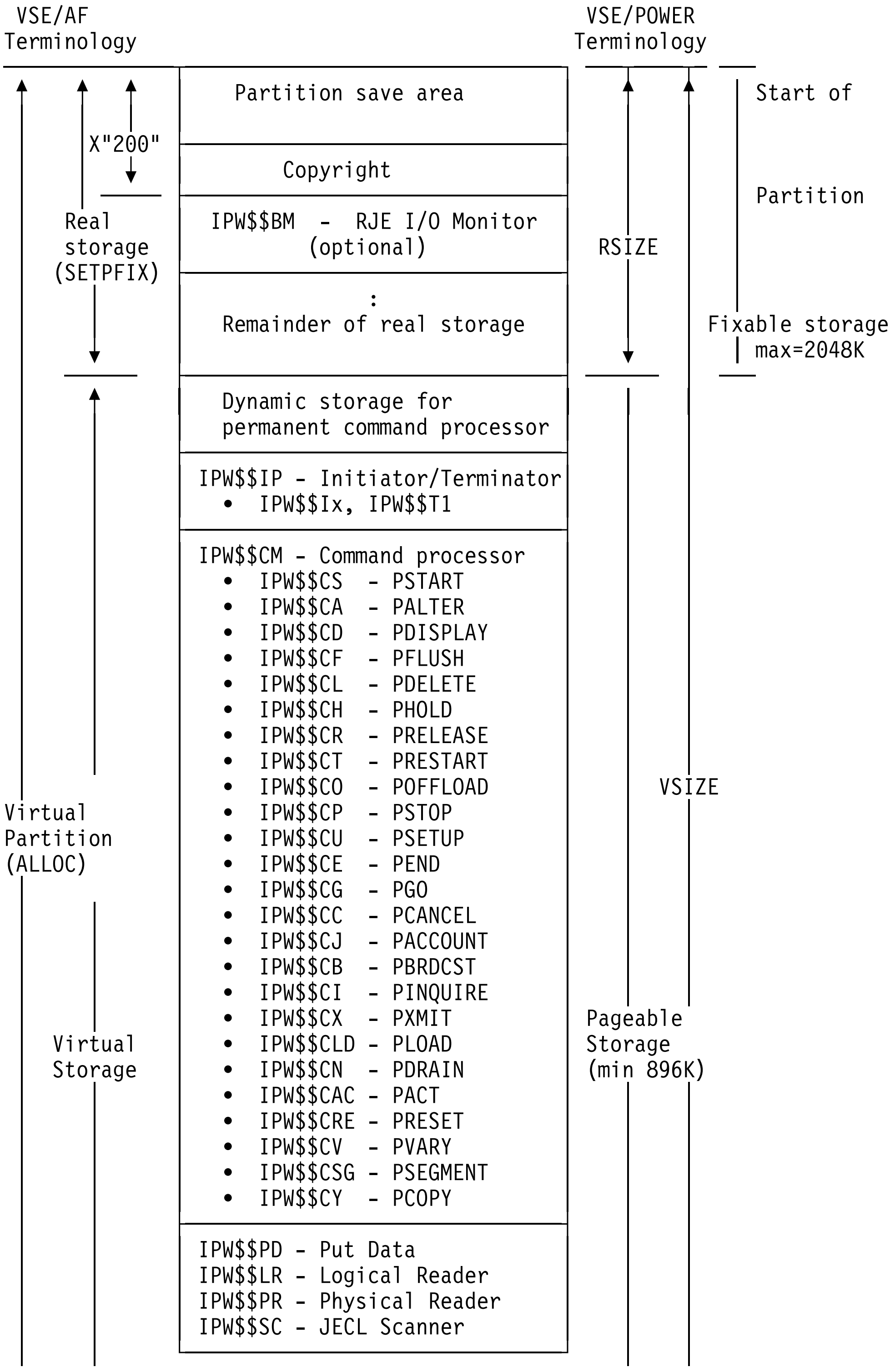
Figure 139. Control Blocks in the SVA Part of VSE/POWER

How to Locate the CAT

Since register 10 always points to the CAT, the pointer to the CAT can be found in the VSE/POWER partition save area. That is register 10 is saved at offset X'14' of the VSE/POWER partition. It also can be found at offset X'5C' (IJBPWR) of the system communication region.

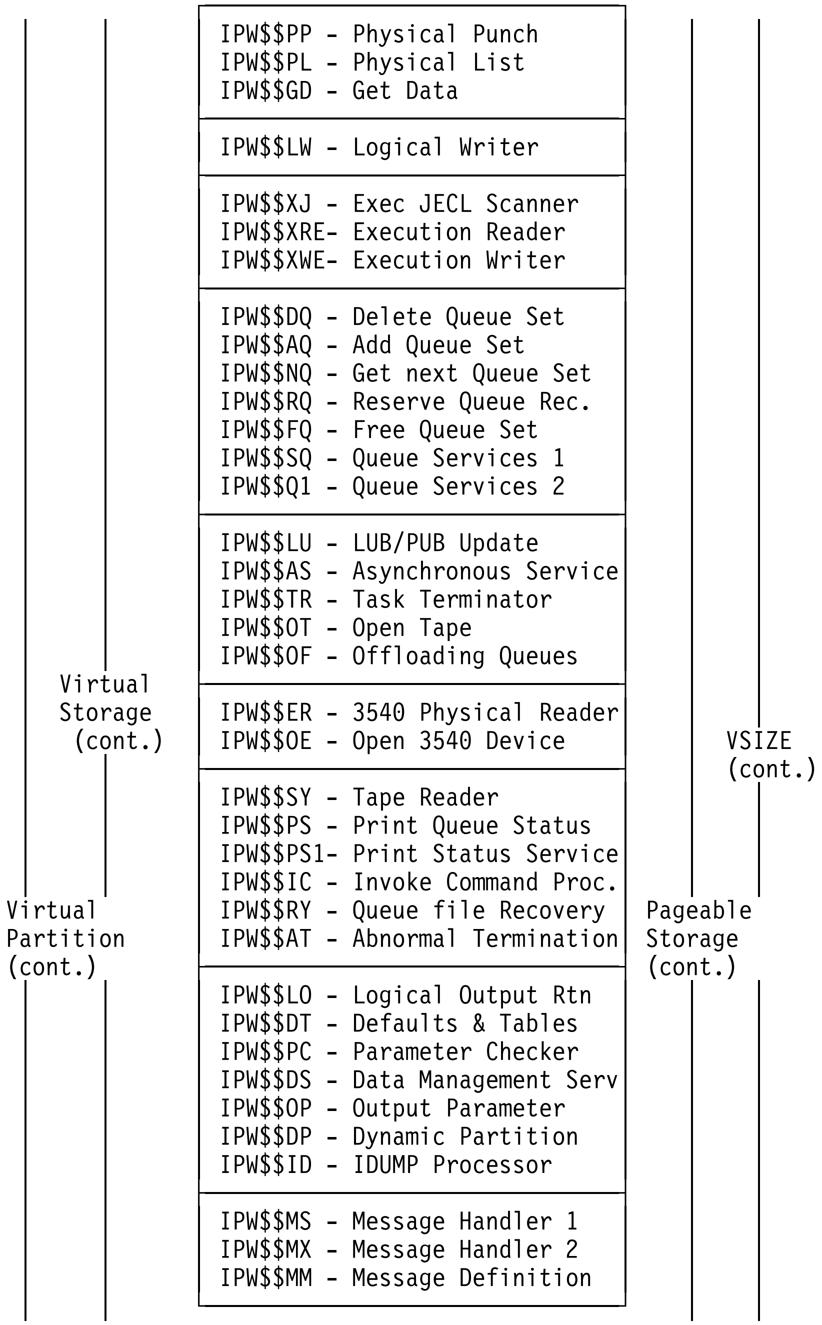
The Storage Layout of the VSE/POWER Partition

The storage layout of the VSE/POWER partition is illustrated in Figure 140.



(cont).

Figure 140 (Part 1 of 4). Storage Layout of VSE/POWER Partition



IPW$$PP - Physical Punch
IPW$$PL - Physical List
IPW$$GD - Get Data

IPW$$LW - Logical Writer

IPW$$XJ - Exec JECL Scanner
IPW$$XRE- Execution Reader
IPW$$XWE- Execution Writer

IPW$$DQ - Delete Queue Set
IPW$$AQ - Add Queue Set
IPW$$NQ - Get next Queue Set
IPW$$RQ - Reserve Queue Rec.
IPW$$FQ - Free Queue Set
IPW$$SQ - Queue Services 1
IPW$$Q1 - Queue Services 2

IPW$$LU - LUB/PUB Update
IPW$$AS - Asynchronous Service
IPW$$TR - Task Terminator
IPW$$OT - Open Tape
IPW$$OF - Offloading Queues

IPW$$ER - 3540 Physical Reader
IPW$$OE - Open 3540 Device

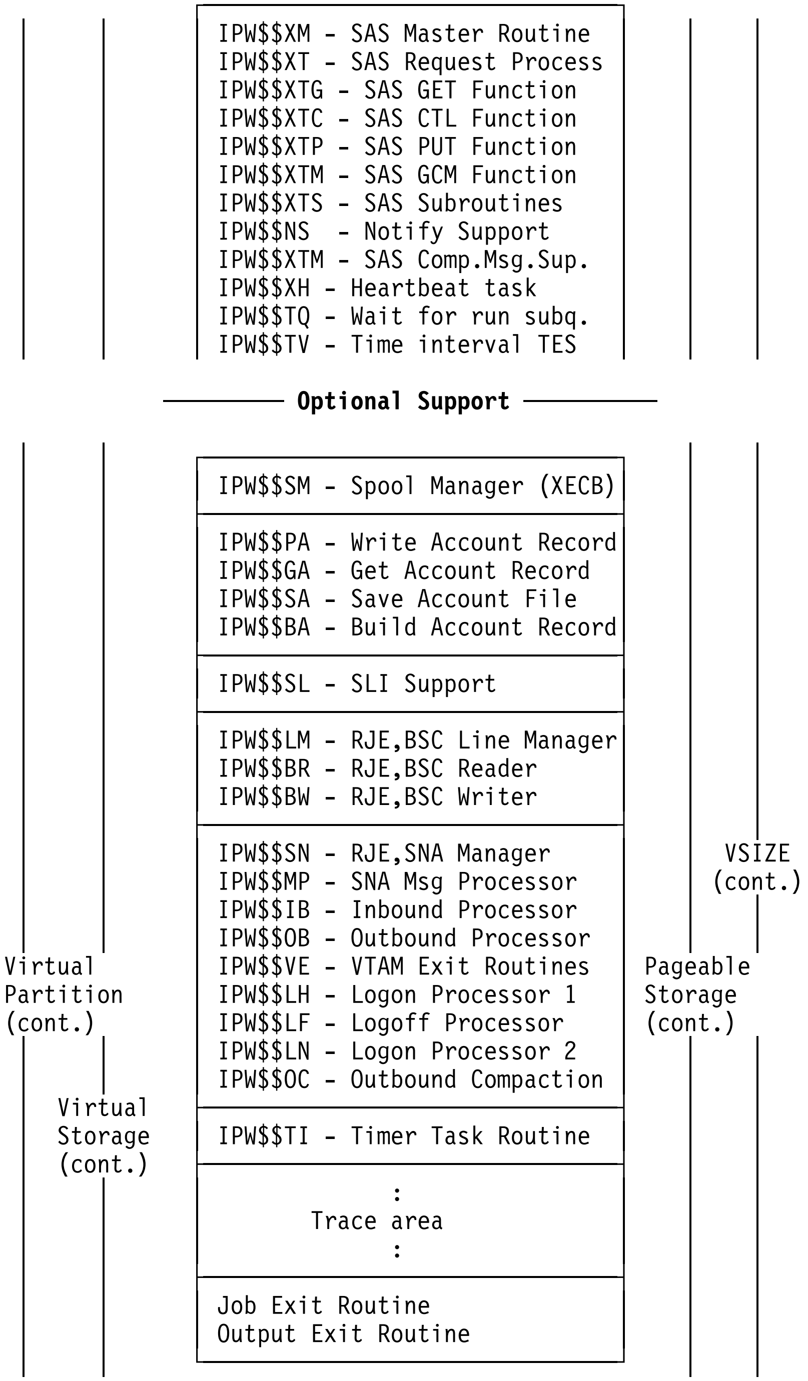
IPW$$SY - Tape Reader
IPW$$PS - Print Queue Status
IPW$$PS1- Print Status Service
IPW$$IC - Invoke Command Proc.
IPW$$RY - Queue file Recovery
IPW$$AT - Abnormal Termination

IPW$$LO - Logical Output Rtn
IPW$$DT - Defaults & Tables
IPW$$PC - Parameter Checker
IPW$$DS - Data Management Serv
IPW$$OP - Output Parameter
IPW$$DP - Dynamic Partition
IPW$$ID - IDUMP Processor

IPW$$MS - Message Handler 1
IPW$$MX - Message Handler 2
IPW$$MM - Message Definition

(cont).

Figure 140 (Part 2 of 4). Storage Layout of VSE/POWER Partition



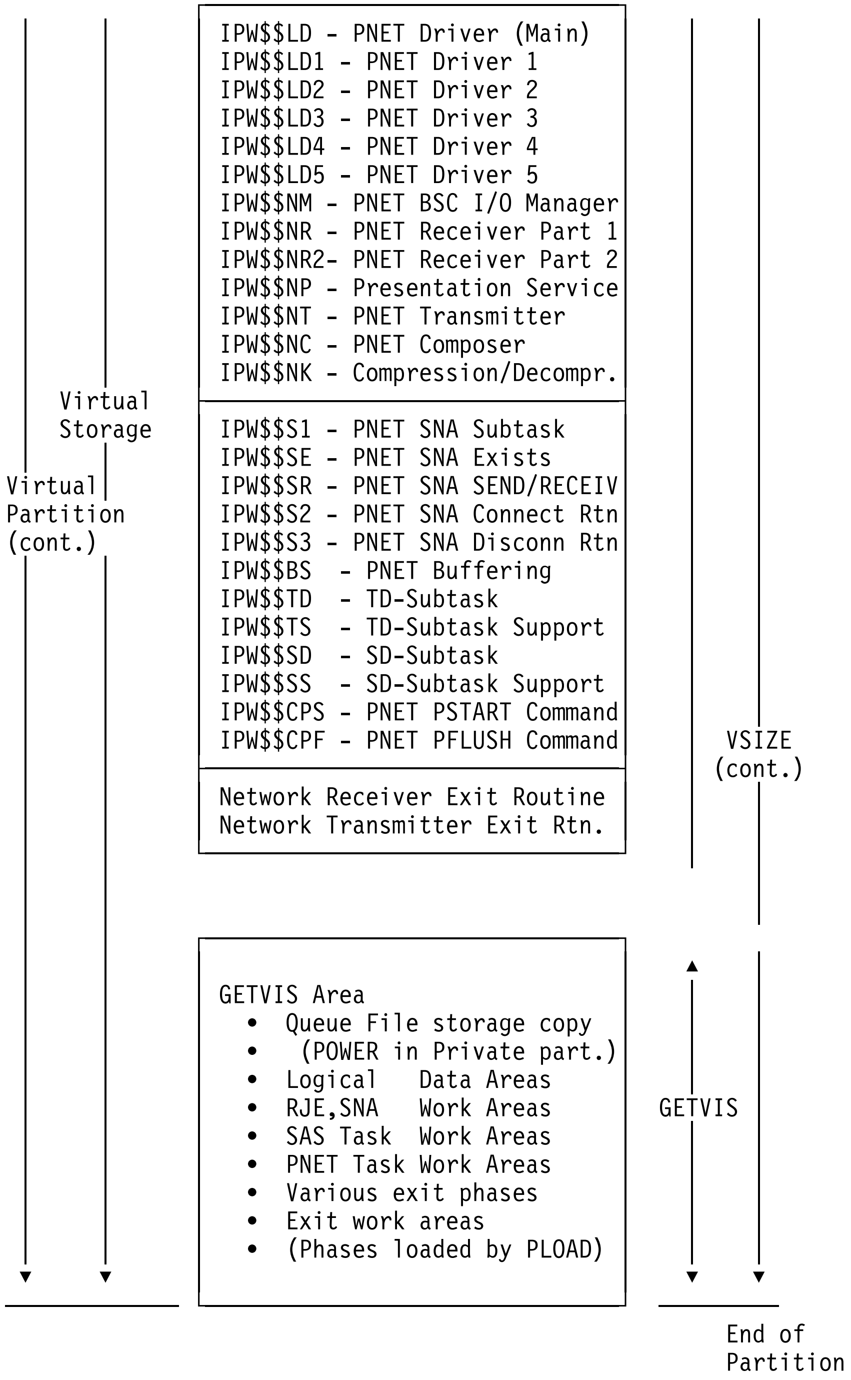
```
IPW$$XM - SAS Master Routine
IPW$$XT - SAS Request Process
IPW$$XTG - SAS GET Function
IPW$$XTC - SAS CTL Function
IPW$$XTP - SAS PUT Function
IPW$$XTM - SAS GCM Function
IPW$$XTS - SAS Subroutines
IPW$$NS  - Notify Support
IPW$$XTM - SAS Comp.Msg.Sup.
IPW$$XH - Heartbeat task
IPW$$TQ - Wait for run subq.
IPW$$TV - Time interval TES
```

Optional Support

```
IPW$$SM - Spool Manager (XECB)

IPW$$PA - Write Account Record
IPW$$GA - Get Account Record
IPW$$SA - Save Account File
IPW$$BA - Build Account Record

IPW$$SL - SLI Support

IPW$$LM - RJE,BSC Line Manager
IPW$$BR - RJE,BSC Reader
IPW$$BW - RJE,BSC Writer

IPW$$SN - RJE,SNA Manager
IPW$$MP - SNA Msg Processor
IPW$$IB - Inbound Processor
IPW$$OB - Outbound Processor
IPW$$VE - VTAM Exit Routines
IPW$$LH - Logon Processor 1
IPW$$LF - Logoff Processor
IPW$$LN - Logon Processor 2
IPW$$OC - Outbound Compaction

IPW$$TI - Timer Task Routine

        :
   Trace area
        :

Job Exit Routine
Output Exit Routine
```

(cont).

Figure 140 (Part 3 of 4). Storage Layout of VSE/POWER Partition



IPW$$LD - PNET Driver (Main)
IPW$$LD1 - PNET Driver 1
IPW$$LD2 - PNET Driver 2
IPW$$LD3 - PNET Driver 3
IPW$$LD4 - PNET Driver 4
IPW$$LD5 - PNET Driver 5
IPW$$NM - PNET BSC I/O Manager
IPW$$NR - PNET Receiver Part 1
IPW$$NR2- PNET Receiver Part 2
IPW$$NP - Presentation Service
IPW$$NT - PNET Transmitter
IPW$$NC - PNET Composer
IPW$$NK - Compression/Decompr.

IPW$$S1 - PNET SNA Subtask
IPW$$SE - PNET SNA Exists
IPW$$SR - PNET SNA SEND/RECEIV
IPW$$S2 - PNET SNA Connect Rtn
IPW$$S3 - PNET SNA Disconn Rtn
IPW$$BS - PNET Buffering
IPW$$TD - TD-Subtask
IPW$$TS - TD-Subtask Support
IPW$$SD - SD-Subtask
IPW$$SS - SD-Subtask Support
IPW$$CPS - PNET PSTART Command
IPW$$CPF - PNET PFLUSH Command

Network Receiver Exit Routine
Network Transmitter Exit Rtn.

GETVIS Area
- Queue File storage copy
- (POWER in Private part.)
- Logical Data Areas
- RJE,SNA Work Areas
- SAS Task Work Areas
- PNET Task Work Areas
- Various exit phases
- Exit work areas
- (Phases loaded by PLOAD)

Figure 140 (Part 4 of 4). Storage Layout of VSE/POWER Partition

The Permanent Area

The permanent area consists only of the RJE/BSC manager (IPW$$BM) if VSE/POWER was generated with RJE/BSC. If RJE/BSC is not generated the area is added to the fixable area.

The Fixable Area

The fixable area consists of the following control blocks:

Description of Use	Storage Descriptor
Initiator/Terminator TCB	I IT
Disk Management Block - Resource Control Fields - Record Control Fields - Master Record Area - Auxiliary Queue Record Area	DMB
Command Processor TCB	O CP
Virtual Storage Control Block	VSCB
Module Control Block (Q) Module Control Block (D)	MCB QFILE MCB DFILE
I/O Buffer Area (Q-file) I/O Buffer Area (D-file)	
Communicator Information Block	CIB
Communicator Information Block 2	CI2
Dynamic Partition Scheduling Control Block	DPCB
Asynchronous Service Work Space	ASWS
EXIT Table	
Account Control Block (optional)	ACCB
Master External Device Control Block	MEDCB

Figure 141. Control Blocks Permanently Allocated in the Fixable Area

These control blocks (Figure 141) are initialized at VSE/POWER startup time (IPW$$IP) and remain in the fixable area until VSE/POWER is terminated. The location of each block is kept in the CAT. Each block has a storage descriptor enabling easy identification in a storage dump.

These blocks (Figure 142 on page 440) are dynamically constructed, depending on the tasks required at any given time. The organization of the blocks relative to each other and the start of the fixable area cannot be truly illustrated. The figure, however, lists those blocks that are eligible to be in the fixable area.

Description of Use	Storage Descriptor
Task Control Block - Task Management Fields - Task Register Save Area - General Task Work Area - Linkage Register Save Area - File Control Words - Command Processor Control Block	TCB CPB
Communicator Information Element	CIE
Communicator Information Block for Job Comp. Message Retrieval	CI2
Linkage Register Save Area	None
Physical Work Space	None
Physical Data Buffer	None
Tape Control Block	TBB
Buffer Control Block	None
Queue Record Area	None
Account Control Block	ACCB
Account Work Space	None
Account Records - Execution Account Record - RJE,BSC Line Account Record - RJE,SNA Session Account Record	None

Figure 142 (Part 1 of 2). Control Blocks Dynamically Allocated in the Fixable Area

Description of Use	Storage Descriptor
RJE/BSC Line Control Block	LCB
RJE/SNA Control Block	SNCB
Remote Message Control Block	MSCB
CCB	None
CCW	None
Diskette Work Space	OEWS
Asynchronous Service Anchor Block	ASAB
Service Request Block	None
Assign/Unassign Work Space	LUWS
TCB Extension Area	None
Print Status Work Area	None
PNET Master Control Block	PNCB
PNET BSC Transmission Buffer	None
PNET SSL Driver Control Block	SDCB
PNET TCP Driver Control Block	TDCB
Node Control Block	NCB
Trace Information Block	TIB
Exit data table	none
VTAM Driver Control Block	VDCB

Figure 142 (Part 2 of 2). Control Blocks Dynamically Allocated in the Fixable Area

The GETVIS Area

The GETVIS area is an extension of the pageable area and is used in its Getvis-24 part for the following purposes:

- Queue File Storage Copy (if no Getvis-31 part is allocated)
- RJE,SNA operation
- 3200/3800 printer setup processing
- Logical data areas associated with each task
- Input buffer for SYSIN tape support
- PNET SNA transmission buffers
- Message queue(s)
- Input/Output buffer for Accounting (FBA only)
- Work area for the various PNET tasks
- Work area for the SAS user task
- Work areas for various exit routines
- Exit phases (loaded via PLOAD)
- VSE/POWER phases (loaded via PLOAD)

and is used in its Getvis-31 part for:

- Queue File Storage Copy, that may even stretch over the 16MB line

The areas (apart from Queue File Storage copy) are allocated by the appropriate tasks when needed and freed when the tasks terminate or when no longer needed.

The GETVIS area is divided into the following pools:

- General pool
- Message pool
- PNET pool
- RJE,SNA pool
- RJE,SNA WACB and compaction table pool

Each pool is aligned at page boundary and consists of an integer number of pages. If a page within a pool becomes empty it is automatically freed by VSE/AF so that the GETVIS storage is available for other pools.

The control blocks which are allocated in the GETVIS area are shown in Figure 143 on page 443.

Description of Area	Storage Descriptor
Storage Copy of Queue File on Disk	None
Account Records - Reader Account Record - List Account Record - Punch Account Record - Execution Account Record - RJE,BSC Line Account Record - Startup Account Record - Transmitter Account Record - Receiver Account Record - Spool Account Record - SAS Connection Account Record	None
Communicator Information Element for Job Comp.Msg. Retrieval	ACIE
FCB Table	
RJE,SNA Session Control Block	SUCB
RJE,SNA Logical Unit Control Block	LUCB
RJE,SNA Logon Request Control Block	LRCB
RJE,SNA Work Area	WACB
RJE,SNA Remote Control Block	RMCB
Network Definition Table	NDT
Network Receiver Work Area	None
Receiver Presentation Work Area	NPWA
Network Transmitter Work Area	None
Composer Work Area	NCWA
PNET Account Record	None
SNA Session Control Block	SSCB
SNA Request Queue Element	SRQE
Command Processor Work Area	----
Spool Environment Block	SPB
SL - Work Area	SLWA
SL - Member Element	None
XT - Work Area	XTWAREA
External Device Control Block	EDCB
Logical Data Area	None
FCB Table	FCBCB
Application Commun. Info Element	ACIE
Asynchronous Service Work Element	ASWE

Figure 143. Control Blocks Dynamically Allocated in the GETVIS Area

Account Control Block (ACCB)

Definition Macro: IPW$DAC

The ACCB is used only by job accounting support. It is used to control account records contained on the account file 'IJAFILE' (SYS000).

The ACCB is initialized for PUT mode. The mode is changed into GET mode when the save account task issues a IPW$OAF macro. The format of the block as printed in a dump is as follows:

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
ACCOUNT CONTROL BLOCK (CKD)					
(0)	0	CHARACTER	16	ACSD	SECTION DESCRIPTOR
(10)	16	BITSTRING	4	ACEB	EVENT CONTROL BLOCK
(14)	20	BITSTRING	4	ACLO	EXTENT LOWER LIMIT
(18)	24	BITSTRING	4	ACHI	EXTENT UPPER LIMIT
(1C)	28	BITSTRING	4	ACLW	LOCKWORD
(20)	32	BITSTRING	4	ACQ32ID	1Q32A MESSAGE ID FOR DOM
(24)	36	BITSTRING	4		UNUSED, KEEP CCW DW ALIGNED
COMMAND CONTROL BLOCK					
(28)	40	BITSTRING	16	ACCB (0)	COMMAND CONTROL BLOCK
(28)	40	BITSTRING	2	ACCT	RESIDUAL COUNT
(2A)	42	BITSTRING	2	ACCM	COMMUNICATION BYTES
(2C)	44	BITSTRING	2	ACST	DEVICE STATUS
(2E)	46	BITSTRING	2	ACLU	LOGICAL UNIT
(30)	48	BITSTRING	1		RESERVED FOR LIOCS
(31)	49	BITSTRING	3	ACCA	CCW REAL-ADDRESS
(34)	52	BITSTRING	1		RESERVED FOR PIOCS
(35)	53	BITSTRING	3	ACCS	CCW ADDRESS IN CSW
(38)	56	ADDRESS	4	ACTB	SAVE ACCOUNT TCB ADDRESS
(3C)	60	BITSTRING	1	ACPB	PUB DEVICE TYPE CODE
(3D)	61	BITSTRING	1	ACDT	DTFPH DEVICE TYPE CODE
(3E)	62	BITSTRING	2		FLAG BYTE 1 & 2
(40)	64	CHARACTER	8	ACPR	BLOCK AND RECORD LENGTH This area consists of the eight-byte control field described below together with the first part of the account record, which describes the VSE/POWER supplied standard prefix. It is used to contain the block length and record length of the account record to be written. Thus account records are formatted as standard variable length records.
VSE/POWER PREFIX FOR ACCOUNT RECORDS (OPTIONAL)					
(48)	72	CHARACTER	16	ACPRF (0)	ACCOUNT RECORD PREFIX
(48)	72	CHARACTER	1	ACPID	SYSTEM ID
(49)	73	CHARACTER	1	ACPRT	RECORD TYPE
(4A)	74	BITSTRING	1		VERSION LEVEL
(4B)	75	CHARACTER	8	ACCMP	COMPONENT ID
(53)	83	CHARACTER	5		RESERVED
DEVICE INFORMATION AND CHARACTERISTICS					
(58)	88	BITSTRING	7	ACSA	CURRENT SEEK ADDR BBCCHHR
(5F)	95	BITSTRING	1		RESERVED
(60)	96	BITSTRING	8	ACCF	COUNT FIELD
(68)	104	BITSTRING	4	ACMC	MAX. ACCOUNT FILE CAPACITY
(6C)	108	BITSTRING	4	ACEC	20 % LIMIT RESIDUAL CAPACITY

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(70)	112	BITSTRING	4	ACAC	CURRENT RESIDUAL CAPACITY
(74)	116	BITSTRING	4	ACMT	MAXIMUM TRACK CAPACITY
(78)	120	BITSTRING	4	ACLC	RESID CAP ON CURRENT TRACK
(7C)	124	BITSTRING	4	AC#T	NUMBER TRACKS/CYLINDER
(80)	128	BITSTRING	2	ACSE	SECTOR VALUES
(82)	130	BITSTRING	2	ACUH	UPPER HEAD
(84)	132	BITSTRING	2	ACDL	TOTAL ACCOUNT RECORD LENGTH
(86)	134	BITSTRING	2	ACPRL	LNGTH OF BLOCK FIELD & PRFX
CHANNEL PROGRAM					
(88)	136	CHAR-ACTER	56	ACCH (0)	CHANNEL PROGRAM
(88)	136	BITSTRING	8	ACSK	SEEK CCW
(90)	144	BITSTRING	8	ACSS	SET SECTOR OR TIC +8 CCW
(98)	152	BITSTRING	8	ACSH	SEARCH ID.EQUAL CCW
(A0)	160	BITSTRING	8	ACTI	TIC -8 CCW
(A8)	168	BITSTRING		ACRW (0)	WCKD CCW'S
(A8)	168	BITSTRING	8	ACWC	.. WRITE COUNT
(B0)	176	BITSTRING	8	ACWD	.. WRITE ACCOUNT DATA
		1.1. 1...		ACRDD	"ACWC" .. RESPECIFY FOR READ DATA
		1.11		ACRCT	"ACWD" .. RESPECIFY FOR READ COUNT
(B8)	184	BITSTRING	8	ACRS	READ SECTOR OR NOT USED
(C0)	192	BITSTRING	16	ACPM	CHAN.PROG MODIFIERS RDATA AND RCOUNT CCW'S
(D0)	208	BITSTRING	4	ACWA	VIRT ADDR WORKSP BUFFER
(D4)	212	BITSTRING	11		UNUSED
(DF)	223	BITSTRING	1	ACFFLG1	FLAG BYTE 1
		1...		ACF1M32A	"X'80'" ..MSG 1Q32A ISSUED
					X'40' ..RESERVED FOR FUTURE USE
					X'20' ..RESERVED FOR FUTURE USE
					X'10' ..RESERVED FOR FUTURE USE
					X'08' ..RESERVED FOR FUTURE USE
					X'04' ..RESERVED FOR FUTURE USE
					X'02' ..RESERVED FOR FUTURE USE
					X'01' ..RESERVED FOR FUTURE USE
(E0)	224	BITSTRING	24	ACAFTI	TIMER ELEMENT FOR 1Q31I
		SIGNED		ACMRL	"2008" .. MAX. ACCOUNT RECORD SIZE
(F8)	248	ADDRESS	1		
		1111 1...		ACLN	"*-ACDS" LENGTH

Account File on an FBA Device

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
ACCOUNT CONTROL BLOCK (FBA)					
(0)	0	CHAR-ACTER	16	AFSDF	SECTION DESCRIPTOR
(10)	16	SIGNED	4	AFEBF	EVENT CONTROL BLOCK
(14)	20	SIGNED	4	AFLOF	LOWER LIMIT (BLOCK NBR)
(18)	24	SIGNED	4	AFHIF	UPPER LIMIT (BLOCK NBR)
(1C)	28	SIGNED	4	AFLWF	LOCKWORD
(20)	32	BITSTRING	4	AFQ32ID	1Q32A MESSAGE ID FOR DOM
(24)	36	BITSTRING	4		UNUSED, KEEP CCW DW ALIGNED
COMMAND CONTROL BLOCK					
(28)	40	BITSTRING	16	AFCBF (0)	COMMAND CONTROL BLOCK
(28)	40	BITSTRING	2	AFCTF	RESIDUAL COUNT
(2A)	42	BITSTRING	2	AFCMF	COMMUNICATION BYTES
(2C)	44	BITSTRING	2	AFSTF	DEVICE STATUS
(2E)	46	BITSTRING	2	AFLUF	EXCP REAL PLUS LUB INDEX
(30)	48	BITSTRING	1		
(31)	49	ADDRESS	3	AFCAF	CCW ADDRESS
(34)	52	BITSTRING	1		

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(35)	53	ADDRESS	3	AFLAF	CCW ADDRESS IN CSW
(38)	56	ADDRESS	4	AFTCB	SAVE ACCOUNT TCB ADDRESS
(3C)	60	BITSTRING	1	AFPBF	PUB DEVICE TYPE CODE
(3D)	61	BITSTRING	1	AFDTF	DTF DEVICE TYPE CODE
		CHARACTER		AFFDF	"C'F'" ...FBA DEVICE
(3E)	62	BITSTRING	1	AFSTATUS	PROCESSING STATUS BYTE
		1...		AFSTOLD	"X'80'" PROCESS CURRENT CI
		.1..		AFACTIV	"X'40'" IPW$$PF WAIT ON FULL FILE
		..1.		AFSFWAIT	"X'20'" IPW$$SF IS IN WFI-STATE
		.... ..1.		AFCOLD	"X'02'" IPW$$IR COLDSTART ACC.FILE
		1		AF80AFF	"X'01'" 80% OF ACCOUNT FILE FULL
(3F)	63	BITSTRING	1		RESERVED
(40)	64	CHARACTER	8	AFPR	BLOCK AND RECORD LENGTH This area consists of the eight-byte control field described below together with the first part of the account record, which describes the VSE/POWER supplied standard prefix. It is used to contain the block length and record length of the account record to be written. Thus account records are formatted as standard variable length records.
	VSE/POWER PREFIX FOR ACCOUNT RECORDS (OPTIONAL)				
(48)	72	CHARACTER	16	AFPRF (0)	ACCOUNT RECORD PREFIX
(48)	72	CHARACTER	1	AFPID	SYSTEM ID
(49)	73	CHARACTER	1	AFPRT	RECORD TYPE
(4A)	74	BITSTRING	1		VERSION LEVEL
(4B)	75	CHARACTER	8	AFCOMP	COMPONENT ID
(53)	83	CHARACTER	5		RESERVED
(58)	88	BITSTRING	1		RESERVED
	Current Address				
(59)	89	BITSTRING	7	AFSAF (0)	CURRENT ADDRESS BRRNNNN
(59)	89	BITSTRING	1	AFSAFB	RESERVED
(5A)	90	BITSTRING	2	AFSAFR	RECORD NUMBER
(5C)	92	BITSTRING	4	AFSAFN	BLOCK NUMBER
(60)	96	SIGNED	4	AFSIC	FREE SPACE IN CURRENT CI
(64)	100	SIGNED	2	AFBPC	NO. OF FBA BLOCKS PER CI
(66)	102	BITSTRING	2		RESERVED
(68)	104	SIGNED	4	AFMCF	MAX. ACCOUNT FILE CAPACITY
(6C)	108	SIGNED	4	AFECF	80% LIMIT, IF REACHED, MSG
(70)	112	SIGNED	4	AFACF	CURRENT RESIDUAL CAPACITY
	CI Description				
(74)	116	BITSTRING	12	AFCIFI (0)	CI DESCRIPTION
(74)	116	BITSTRING	4	AFCIF	LENGTH 1 CONTROL INTERVAL
(78)	120	ADDRESS	4	AFWAF	VIRTUAL ADDR. OF WORKSPACE
(7C)	124	ADDRESS	4		RESERVED
(80)	128	ADDRESS	4	AFWASA	VIRT-AD CI-AREA SAVE-ACC
(84)	132	SIGNED	4	AFSACB SAVE-ACCOUNT CURRENT BLOCK	
	Account Record Parameter				
(88)	136	BITSTRING	8	AFPARM (0)	ACCOUNT RECORD PARAMETERS
(88)	136	SIGNED	4	AFPARML	LENGTH
(8C)	140	ADDRESS	4	AFPARMA	ADDRESS
(90)	144	SIGNED	2	AFBLF	FBA BLOCKSIZE
		SIGNED		AFCISZ	"2048" .. ACCOUNT FILE CI SIZE
(92)	146	SIGNED	2		RESERVED

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(94)	148	SIGNED	4		RESERVED
LOCATE CONTROL WORD					
(98)	152	BITSTRING	8	AFLMF (0)	LOCATE CONTROL WORD
(98)	152	BITSTRING	1	AFOBF	OPERATION BYTE
		1		AFWOF	"X'01'" .. WRITE DATA
		.... .11.		AFROF	"X'06'" .. READ DATA
(99)	153	BITSTRING	1	AFRCF	NOT USED
(9A)	154	SIGNED	2	AF#BF	NBR OF BLOCKS TO PROCESS
(9C)	156	SIGNED	4	AFRDF	REL. DISPL. OF BLOCK
EXTENT DESCRIPTION BLOCK					
(A0)	160	BITSTRING	16	AFEDF (0)	EXTENT DESCRIPTION BLOCK
(A0)	160	BITSTRING	1	AFMBF	MASK BYTE
		11..		AFPWF	"X'C0'" ...PERMIT ALL WRITE CMMD
		.... .1..		AFPDF	"X'04'" ...PERMIT DIAGNOSTIC CMMD
		.1..		AFIWC	"X'40'" ...INHIBIT WRITE CMNDS
(A1)	161	BITSTRING	3		RESERVED MUST BE ZERO
(A4)	164	SIGNED	4	AFBBF	PHYS. ADDR. FIRST BLOCK
(A8)	168	SIGNED	4	AFFBF	REL. DISPL. FIRST BLOCK
(AC)	172	SIGNED	4	AFLBF	REL. DISPL. LAST BLOCK
(B0)	176	BITSTRING	8		RESERVED FOR FUTURE USE
CHANNEL COMMAND WORD					
(B8)	184	BITSTRING	24	AFCHF (0)	CHANNEL PROGRAM
(B8)	184	BITSTRING	8	AFDFF (0)	DEFINE EXTENT CCW
(B8)	184	BITSTRING	1		DEFINE EXTENT COMMAND CODE
(B9)	185	ADDRESS	3		EXTENT DESCRIPTION ADDR.
(BC)	188	BITSTRING	2		FLAGS
(BE)	190	SIGNED	2		COUNT
(C0)	192	BITSTRING	8	AFLCF (0)	LOCATE CCW
(C0)	192	BITSTRING	1		LOCATE COMMAND CODE
(C1)	193	ADDRESS	3		LOCATE WORD ADDR.
(C4)	196	BITSTRING	2		FLAGS
(C6)	198	SIGNED	2		COUNT
(C8)	200	BITSTRING	8	AFRWF (0)	READ OR WRITE CCW
(C8)	200	BITSTRING	1	AFRWFO	OPERATION CODE
(C9)	201	ADDRESS	3	AFRWFA	DATA-ADDRESS
(CC)	204	BITSTRING	1	AFRWFF	FLAGS
(CD)	205	BITSTRING	1		RESERVED
(CE)	206	BITSTRING	2	AFRWFL	DATA LENGTH
		.1.. ...1		AFWRITE	"X'41'" CCW WRITE COMMAND
		.1.. ..1.		AFREAD	"X'42'" CCW READ COMMAND CODE
		.11. .1..		AFRDCO	"X'64'" READ DEVICE CHARACTERISTIC
		.1.. ..11		AFLOCO	"X'43'" LOCATE OP-CODE
		.11. ..11		AFDEFO	"X'63'" DEFINE EXTENT OPCODE
(D0)	208	BITSTRING	7		UNUSED
(D7)	215	BITSTRING	1	AFFFLG1	FLAG BYTE 1
		1...		AFF1M32A	"X'80'" ..MSG 1Q32A ISSUED
					X'40' ..RESERVED FOR FUTURE USE
					X'20' ..RESERVED FOR FUTURE USE
					X'10' ..RESERVED FOR FUTURE USE
					X'08' ..RESERVED FOR FUTURE USE
					X'04' ..RESERVED FOR FUTURE USE
					X'02' ..RESERVED FOR FUTURE USE
					X'01' ..RESERVED FOR FUTURE USE
(D8)	216	BITSTRING	24	AFAFTI	TIMER ELEMET FOR 1Q31I
(F0)	240	ADDRESS	1		
		1111 1...		AFLNF	"*-AFSDF" LENGTH DESCRIPTOR

How to Locate: Refer to Figure 152 on page 731 in Chapter 6, “Diagnostic Aids.”

Accounting Record Layouts

The various account records layouts:

- Account Record - Execution
- Account Record - List
- Account Record - Network
- Account Record - Punch
- Account Record - Reader
- Account Record - Receiver
- Account Record - RJE/BSC
- Account Record - RJE/SNA
- Account Record - System Startup
- Account Record - Transmitter
- Account Record - Spool Access Support
- Account Record - Spool Account Record

are to be found in the *VSE/POWER Application Programming Guide*.

Assign/Unassign Work Space

DSECTname: LUWKDS

The work space is primarily used as a register save area and to contain the address to the SETPRT parameter list when a 3800 printer is being unassigned and asynchronous service is invoked to set up the printer with the system/hardware defaults.
The DSECT is defined in IPW$$LU.

Bytes	Label	
Hex.	of Field	Description/Function
00-01	LUWKPIK	PIK of partition concerned
02-03	LUWKPHY	Physical unit number (PUB index)
04-05	LUWKCHN	Channel unit number
06-07	LUWKLOG	Logical unit number (CCB format)
08	LUWKMAC	Failing macro id
09-0B		Reserved for future use
0C-0D	LUWKPUB	PUB index save area
0E-0F		Reserved for future use
10-13	LUWKSLU	Pointer to system LUB
14-17	LUWKPLU	Pointer to programmer LUB
18-1F	LUWKGR	Save area for registers 14-15. Used when another function is invoked.
20-57	LUWKSV	Temporary register save area for the interface between functions.
58-5B	LUWKSP	Address to SETPRT parameter list

Asynchronous Service Anchor Block (ASAB)

Definition Macro: IPW$DAB

The asynchronous service anchor block contains the queue pointers for all service requests to be performed by the service subtask. Storage for the anchor block is acquired the first time a VSE/POWER initialization task issues the ATTACH request. The anchor block exists as long as VSE/POWER is active.

Bytes Hex.	Label of Field	Description/Function
00-0F	ABSD	Storage descriptor (ASWS)
10-13	ABADR1	Address of SETPRT routine (IJVSRPDV) in SVA
14-1B		Reserved for future use
1C-1F	ABLCK	Lock word

- The following fields are used by the IDUMP service

Bytes Hex.	Label of Field	Description/Function
20-23	ABDECB	IDUMP subtask ECB
24-27	ABCECB	IDUMP communication ECB
28-2B	ABDPFR	Address of first request in queue
2C-2F	ABCPLR	Address of last request in queue
30-33	ABDSAP	Address of active SRB
34-37		Reserved for future use

- The following fields are used by the library access service

Bytes Hex.	Label of Field	Description/Function
38-3B	ABRECB	Library access subtask ECB
3C-3F	ABRECB2	Library access subtask termination ECB
40-43	ABRPHD	pointer to first request in queue
44-47	ABRPTL	pointer to last request in queue
48-4B	ABLSAP	Address of active SRB
4C	ABLFLG1	Pointer to active SRB
	ABLF1DIP	X'80' - LS Subtask Detach in Process
4D-4F		Reserved for future use
50-53	ABICCF	Entry point address of VSE/ICCF LIB module
54-55	ABRPBL	VSE/ICCF process buffer length
56	ABRFL1	Flag byte 1
	ABRIOP	X'40' - VSE/ICCF open done
57	ABRFL2	Flag byte 2
	ABRPNF	X'20' - VSE/ICCF module not found
58-59	ABLIBTK	TIK of Libr Subtask
5A-5B		Reserved for future use
	ABLN	Length of control block

Asynchronous Service Control Section (ASCB)

Definition Macro: IPW$DAB

The asynchronous control section consists of the TQE (timer queue element), the wait-for-multiple-list and the asynchronous service ecb, which is posted by a supervisor detach. The asynchronous work section is located within real storage to be able to use the VSE/POWER macros.

Offset Hex	Type	Len	Name (Dim)	Description
(0)	BITSTRING	24	ASCTQE	TQE FOR ASYNCHR. SERVICE
(18)	BITSTRING	4	ASCECB	ASYNCHR. SERVICE ECB
(1C)	SIGNED	2	ASCMLST (0)	WAIT-FOR-MULTIPLE-LIST
(1C)	BITSTRING	4	ASCTQECB	..ADDRESS OF TQE ECB
(20)	BITSTRING	4	ASCASECB	..ADDRESS OF AS. SERV. ECB
(24)	BITSTRING	1	ASCENDL	..END OF WFM-LIST
(25)	BITSTRING	3		RESERVED FOR FUTURE USE
	..1. 1...		ASCLN	"*-ASCDS" LENGTH OF CONTROL SECTION

How to Locate: The address of this control block is located in the field ASWEECB of the Asynchronous Service Work Element

Asynchronous Service Work Element (ASWE)

Definition macro: IPW$DAB

The asynchronous service work element is acquired and formatted by the asynchronous service function routine. One element exits for each asynchronous service subtask currently active. The element contains important information for the subtask, such as the register save area.

Offset Hex	Type	Len	Name (Dim)	Description
	ASYNCHRONOUS SERVICE WORK ELEMENT			
(0)	CHARACTER	8	ASWENAM	SUBTASK NAME, ALWAYS IPW$$AS
(8)	DBL WORD	8	ASWEREG (14)	SUBTASK'S REGISTERS
(78)	CHARACTER	8	ASWENAM2	SUBTASK NAME, ALWAYS IPW$$AS
(80)	BITSTRING	216	ASWEABN	PSW & ABN REGISTER SAVEAREA
(158)	ADDRESS	4	ASWEECB	POINTER TO ASCS
(15C)	ADDRESS	4	ASWESRB	ADDRESS OF ASSOCIATED SRB
(160)	ADDRESS	4	ASWETCB	ADDRESS OF ASSOCIATED TCB
(164)	BITSTRING	1	ASWEFLG	FLAG BYTE 1
	1...		ASWEFMI	"X'80'" .. 1QA0I MSG ALREADY ISSUED
	.1..		ASWEDUMP	"X'40'" .. $AT IN DUMP PROCESSING
(165)	BITSTRING	3		RESERVED FOR FUTURE USE
(168)	SIGNED	2	ASWEGDL	LIST LENGTH
(16A)	CHARACTER	8		PHASENAME
(172)	BITSTRING	3		
(175)	ADDRESS	1		N
(176)	BITSTRING	1		PHASE INFORMATION
	LOCAL SAVE AREA FOR IPW$IDM REQUESTS			
(194)	SIGNED	4	ASWST0E	REGISTER RE SAVE AREA
(198)	SIGNED	4	ASWST0F	REGISTER RF SAVE AREA
(19C)	SIGNED	4	ASWST00	REGISTER R0 SAVE AREA
(1A0)	SIGNED	4	ASWST01	REGISTER R1 SAVE AREA
(1A4)	SIGNED	4	ASWST02	REGISTER R2 SAVE AREA
(1A8)	SIGNED	4	ASWSTID	POINT TO SUBTASK IDENTIFIC.
(1AC)	BITSTRING	4		UNUSED
(1B0)	BITSTRING	16	ASWECCB	CCB AREA FOR NOP
(1E0)	BITSTRING	8	ASWECCW	CCW AREA FOR NOP
(1E8)	BITSTRING	64	ASWEAMFG	WORK AREA FOR LOAD/GETVCE
	EXPRESSION		ASWELEN	"*-ASWEDS" LENGTH OF WORK ELEMENT

Buffer Control Word (BCW)

The buffer control words are used to describe each piece of real storage acquired by real storage management. A buffer control word precedes the storage area, which can be either in use or free, and contains the length of the following storage area and the storage owner, if applicable.

Bytes Hex.	Label of Field	Description/Function
00-01	BCWPRV	Length of previous buffer. This halfword contains the binary length divided by 32 of the immediately-preceding storage buffer. If the buffer is in use its length is stored in two complement forms. If the buffer is not in use its length is stored in normal form. If the present buffer is the first in the fixable area the word is set to binary zeros.
02-03	BCWCUR	Length of next buffer. This halfword contains the binary length divided by 32 of the present storage buffer, that is, the buffer which immediately follows this buffer control word in storage. If the buffer is in use its length is stored in two complement forms. If the buffer is not in use its length is stored in normal form. If the preceding buffer is the last in the fixable area the word is set to binary zeros.
04-07	BCWOWN	Owner (TCB virtual address) of next buffer. This fullword contains the address of the TCB belonging to the task which issued the request for buffer space. If the field contains zeros, the storage is owned by the VSE/POWER system. If a TCB is contained in the buffer, the owner address is that of the task which built the TCB.

Buffer Layout

The control of buffers for PNET is done by buffer management. The layout of the buffers as provided by the function is shown below.

Definition Macro: IPW$DVD BUF=YES

```
Bytes       Label
Hex.        of Field    Description/Function
-----------------------------------------------------------------------

• Buffer Header Common Part

000-003     BUFNEXT     Next buffer in chain
004-007     BUFREAL     Real (pfixed) address of buffer (BSC)
008-00B     BUFOWN      Address of related TCB
00C-00F     BUFNCBA     Address of related NCB
010-011     BUFDATL     Count of bytes to send or received
012-013     BUFSZ       Buffer size (excluding header)
014         BUFSTAT     Status flag
            BUFFREE     X'80' - Release buffer on send complete
            BUFPOST     X'40' - Post task when buffer sent
            BUFRRPL     X'20' - Buffer contains response RPL (SNA)
            BUFBCBI     X'10' - Send 'ignore BCB'  (BSC)
            BUFTERM     X'08' - Terminating buffer (BSC)
            BUFSTCH     X'04' - Status change requested
            BUFQPRI     X'02' - Place buffer in priority queue
015         BUFEST1     Data stream status
            BUFRIF      X'80' - RIF sent/received
            BUFPGR      X'40' - PGR sent/received or
                                receiver cancel sent/received
            BUFPRJ      X'20' - NPGR sent/received
            BUFEOF      X'10' - EOF sent/received
            BUFADS      X'08' - Abort transmission
            BUFCMC      X'04' - EOT sent/received
016-017                 Reserved

• Data Portion BSC

018-020     BUFDATA     Data portion of BSC TP buffer
018-019     BUFSTRT     Transmission control bytes
01A         BUFBCB      Block control byte
            BUFMLIC     X'80' - MLI control bit
            BUFBRES     X'20' - Reset expected block sequence CNT
            BUFBBYB     X'10' - Bypass block sequence validation
01B-01C     BUFFCS      Function control sequence
01D         BUFRCB      Record control byte
01E         BUFSRCB     Subrecord control byte
01F         BUFSCB      String control byte
020         BUFEOB      End-of-block RCB
016-017                 Reserved
```

- Data Portion SNA

018-01A	BUFRIDD	Decompressed RID of 1st record
01B		Unused
01B-083	BUFRPL	Space for VTAM RPL
084-087	BUFDATAS	Data portion of buffer SNA
084	BUFSCBS	SCB
085	BUFRCBS	RCB
086	BUFSRCBS	SRCB
087	BUFRIDL	Length
088	BUFEOBS	End of RU SCB (may be beyond RU)

Cancel Codes of VSE/POWER

Figure 144 shows the VSE/POWER cancel codes that appear in several VSE/POWER records.

Cancel Code	Condition
X'10'	Normal end of VSE/POWER job or task (see Note 1).
X'20'	PCANCEL was issued.
X'30'	PSTOP command was issued (see Note 2).
X'40'	PFLUSH command was issued.
X'50'	PDELETE was issued.
X'60'	VSE/POWER job was flushed via RDREXIT.
X'70'	VSE/POWER job canceled due to I/O error.
X'80'	Job or output transmission canceled due to cancelling of PNET network receiver.
X'90'	SAS quit request
X'A0'	Processing was canceled due to severe error (SAS only)
X'B0'	SAS GET close request.
X'C0'	Processing was terminated due to SOD condition (SAS).
X'D0'	Processing was terminated due to printing/punching failed.

Figure 144. Cancel Codes of VSE/POWER

Notes:

1. Although no abnormal VSE/POWER termination occurred, the VSE/AF jobs associated with the queue entry could have been canceled via VSE/AF.
2. The PSTOP cancel code is not stored in an account record if the EOJ option was specified in the PSTOP command.

Channel Command Word (CCW)

Definition Macro: IPW$DCW

The layout of a Channel Command Word is shown below.

Bytes Hex.	Label of Field	Description/Function
00	CWCC	Command code
01-03	CWDA	Data address
04	CWFL	Chain byte
	DC	X'80' = Data chaining
	CC	X'40' = Command chaining
	SLI	X'20' = Suppress incorrect length
05	CWRE	Reserved
06-07	CWCT	Data length field

- General Flags

	NOP	X'03' = NOP command
	PSKP	X'8B' = Skip to channel-one-flag
	EOPR	X'FE' = End of page reached (internal)

Command Control Block (CCB)

Definition Macro: IPW$DCB

The layout of a Command Control Block is shown below.

Bytes	Label	
Hex.	of Field	Description/Function
00-01	CBCT	Residual count
02	CBC1	First communication byte
	UIO	X'20' = unrecoverable I/O error
	AUIO	X'10' = accept unrecoverable I/O error
	RODC	X'08' = return on data check
	WDE	X'04' = wait for device end
03	CBC2	Second communication byte
	CCR	X'01' = command chain retry option
	CHN9	X'02' = channel 9 overflow
04	CBSD	Device status byte
		X'80' = attention
		X'10' = busy
		X'08' = channel end
		X'04' = device end
	UNCK	X'02' = unit check
	UE	X'01' = unit exception
05	CBSC	Channel status byte
06	CBLC	LUB class
	EXR	X'80' = EXCP real
	PRU	X'01' = programmer unit
07	CBLN	LUB number within class
08	CBLI	LIOCS communication byte
09-0B	CBCA	CCW address
0C	CBPI	PIOCS communication byte
		X'01' = CCW Format 1 is used
0D-0F	CBCS	CCW address in CSW
10	CBNX	First entry outside CCB

- General Flags

	SID	X'20' = sense information desired

Command Processor Work Area (CPWA)

This area contains addresses and information that are used in communication among the various command-processing modules and the root phase. Each command processor task is equipped with such a work area. The work area for the permanent command processor is placed in the first part of the pageable area while the work area for a temporary command processor task is dynamically acquired when needed by means of the IPW$RSV macro instruction.

Definition Macro: IPW$DCP

How to Locate: The permanent command processor task work area is located in the 1st page of the pageable storage (PAVA). Refer to Figure 15 on page 40.
A temporary command processor task work area storage location is stored register 6 of the task.

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
COMMAND PROCESSOR WORK AREA (CPWA)					
(0)	0	STRUCTURE	0	CPWADS	DUMMY SECTION WORKAREA
(0)	0	CHARACTER	32	CPWSD (0)	STORAGE DESCRIPTOR
(0)	0	CHARACTER	28		
(1C)	28	ADDRESS	4	CPWTPTR	POINTER TO TCB OWNING CP WORKAREA
SUBROUTINE ADDRESS TABLE THE FOLLOWING TABLE CONTAINS ADDRESSES TO SUBROUTINES USED BY VARIOUS COMMAND PROCESSORS.					
(20)	32	ADDRESS	4	MSG	ENTRY POINT ADDRESS OF MSG RTN
(24)	36	ADDRESS	4	RELTOABS	ENTRY POINT ADDRESS OF CONVERT RTN
(28)	40	ADDRESS	4	TCBSCAN	EP ADDRESS OF TCBSCAN SUBROUTINE
(2C)	44	ADDRESS	4	BINTODEC	EP ADDRESS OF CONVERT TO BINARY RTN
(30)	48	ADDRESS	4	VPARTID	EP ADDR OF VERIFY PARTITION ID RTN
(34)	52	ADDRESS	4	VTASKID	EP ADDR OF VERIFY TASK ID RTN
(38)	56	ADDRESS	4	VQUEUEID	EP ADDR OF VERIFY QUEUE ID RTN
(3C)	60	ADDRESS	4	QRINSPCT	EP ADDR OF QUEUE REC INSPECT RTN
(40)	64	ADDRESS	4	ATTACH	EP ADDR OF ATTACH SUBROUTINE
(44)	68	ADDRESS	4	ASSIGN	EP ADDR OF ASSIGN SUBROUTINE
(48)	72	ADDRESS	4	INVDEV	EP ADDR OF INVDEV SUBROUTINE
(4C)	76	ADDRESS	4	CLASS	EP ADDR OF CLASS SUBROUTINE
(50)	80	ADDRESS	4	FORMAT	EP ADDR OF FORMAT OPERAND RTN
(54)	84	ADDRESS	4	UNASSGN	EP ADDR OF UNASSGN SUBROUTINE
(58)	88	ADDRESS	4	VERCAUTH	EP ADDR OF VER CMND AUTHOR
(5C)	92	ADDRESS	4	VERKYWOP	EP ADDR OF VER KEYWORD OP'S
(60)	96	ADDRESS	4	AOPCHK	EP ADDR OF "AOPCHK" ROUTINE SCHUPPEN
(64)	100	ADDRESS	4	QRDISPCT	EP ADDR OF DIRECT INSPECTION
(68)	104	ADDRESS	4	(2)	RESERVED FOR FUTURE USE SCHUPPEN
(70)	112	ADDRESS	4	CPCLTAB	ADDR OF INDEX TABLE FOR CLASS ENTRY
COMMUNICATIONAREA					
(74)	116	CHARACTER	8	SWITCHES (0)	SWITCHES USED BY VARIOUS PROCESSORS
(74)	116	BITSTRING	1	SWFLAG1	FLAG BYTE 1
		1...		SWSUPPR	"X'80'" .. TURNED ON BY CALLER OF 'BINTODEC' SUBROUTINE TO INDICATE THAT SUPPRESSION OF LEADING ZEROS IS RE- QUESTED.
		.1..		SWFOUND	"X'40'" .. TURNED ON BY VARIOUS COMMAND PROCESSORS WHEN THEY FOUND A TCB OR QUEUE RECORD OF THE KIND THEY ARE LOOKING FOR
		..1.		SWSTART	"X'20'" .. TURNED ON BY THE PSTART PROCESSOR WHEN A DORMANT PARTITION SHOULD REACTIVATED
(75)	117	BITSTRING	1	SWFLAG2	FLAG BYTE 2

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
		1...		SWREPLY	"X'80'" .. TURNED ON BY CALLER OF MESSAGE ROUTINE TO INDICATE ANSWER REQUIRED
		.1..		SWAUTOST	"X'40'" .. TURNED ON TO INDICATE THAT THE INITIATOR TASK MUST BE PROMPTED TO SUPPLY DEVICE ADDR TO BE SPOOLED
		..1.		SWERROR	"X'20'" .. TURNED ON WHEN AN ERROR MESSAGE SHOULD BE ISSUED TO CENTRAL OPERATOR EVEN WHEN AUTOSTART IS IN PROGRESS
		...1		SWPOALL	"X'10'" .. TURNED ON BY THE POFFLOAD PROC. WHEN EITHER THE ENTIRE XMIT QUEUE OR ALL QUEUES SHOULD BE OFFLOADED
		.... 1...		SWIGNORE	"X'08'" .. TURNED ON BY CALLER OF "CHECK ADDITIONAL OPERANDS" TO INDICATE TYPE OF MESSAGE TO BE ISSUED
		.... .1..		SWF2TST	"X'04'" .. TURNED ON BY IPW$$CD TO SUPPRESS MSG 1R91I IN ADD. OPERANDS CHECK RTN.
(76)	118	BITSTRING	1	SWFLAG3	FLAG BYTE 3
		1...		SWINDEV	"X'80'" .. INDICATES THAT DEVICE TYPE DESIGNATED IN PSTART COMMAND IS IN- CONSISTENT WITH TASK TYPE.
		.1..		SWDELAY	"X'40'" .. INDICATES THAT A WARNING MESSAGE MUST BE DELAYED, TO AVOID DIS- APPEARING OF THE APPROPRITATE TCB
		..1.		SWDEL	"X'20'" .. TURNED ON BY PALTER PROCESSOR WHEN QUEUE SET TO BE ALTERED MUST BE DELETED AND ADDED LATER ON TO CLASS CHAIN RATHER THAN REWRITING OF THE QUEUE RECORD.
		...1		SWNOCCO	"X'10'" .. TURNED ON BY PALTER PROCESSOR WHEN OTHER ATTRIBUTES THAN # OF COPIES SHOULD BE CHANGED.
		.... 1...		SWUSER5	"X'08'" .. TURNED ON BY CALLING ROUTINE TO INDICATE THAT REG 5 SHOULD BE USED AS TCB POINTER.
		.... .1..		SWF3CNC	"X'04'" .. TURNED ON BY IPW$$CS MSG FORCES IPW$CNC IN UNATTENDED SYSTEM
(77)	119	BITSTRING	1	SWFLAG4	FLAG BYTE 4
		1...		SWMSG	"X'80'" .. TURNED ON BY PDISPLAY PROCESSOR IF MESSAGES ARE TO BE SUPRESSED
		.1..		SWPBCST	"X'40'" .. TURNED ON BY PBRDCST PROCESSOR IF MESSAGES HAVE TO BE TRUNCATED
		..1.		SWF4ALL	"X'20'" .. TURNED ON BY PINQUIRE/PSTOP IF 'ALL' IS SPECIFIED OR IF A GROUP OF SIMILAR RESOURCES IS INQUIRED: PNET(BSC|CTC|SNA|TCP) OR RJE(BSC|CTC) OR DEVICES
		...1		SWF4NOTH	"X'10'" .. TURNED ON BY PINQUIRE PROCESSOR IF NOTHING TO DISPLAY
		.... 1...		SWF4NMM	"X'08'" .. TURNED ON BY PDISPLAY PROCESSOR IF MSG NOT TO BE MODIFIED
		.... .1..		SWF4DNC	"X'04'" .. TURNED ON BY PDISPLAY PROCESSOR IF MSG NOT TO BE COMPR'ED
(78)	120	BITSTRING	1	SWFLAG5	FLAG BYTE 5
		1...		SWF5PBSC	"X'80'" .. TURNED ON BY PINQUIRE ALL OR PINQUIRE PNET OR PINQUIRE PNETBSC
		.1..		SWF5PCTC	"X'40'" .. TURNED ON BY PINQUIRE ALL OR PINQUIRE PNET OR PINQUIRE PNETCTC
		..1.		SWF5PSNA	"X'20'" .. TURNED ON BY PINQUIRE ALL OR PINQUIRE PNET OR PINQUIRE PNETSNA
		...1		SWF5PTCP	"X'10'" .. TURNED ON BY PINQUIRE ALL OR PINQUIRE PNET OR PINQUIRE PNETTCP
		.... 1...		SWF5RBSC	"X'08'" .. TURNED ON BY PINQUIRE ALL OR PINQUIRE RJE OR PINQUIRE RJEBSC
		.... .1..		SWF5RSNA	"X'04'" .. TURNED ON BY PINQUIRE ALL OR PINQUIRE RJE OR PINQUIRE RJESNA
		.... ..1.		SWF5DEV	"X'02'" .. TURNED ON BY PINQUIRE ALL OR PINQUIRE DEVICES

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
		1		SWF5PSSL	"X'01'" .. TURNED ON BY PINQUIRE ALL OR PINQUIRE PNET OR PINQUIRE PNETSSL
(79)	121	CHARACTER	3		RESERVED FOR FUTURE USE
REGISTER SAVE AREAS					
(7C)	124	SIGNED	4	CPWRRS (12)	REGISTER SAVEAREA FOR ROOTPHASE
(AC)	172	SIGNED	4	CPWCRS (12)	REGISTER SAVE AREA FOR COMMAND PROC
(DC)	220	SIGNED	4	CPWSRS (12)	REGISTER SAVE AREA FOR SUBROUTINES
(10C)	268	SIGNED	4	CPWFRS (12)	REGISTER SAVE AREA FOR FORMAT SUBROUTINE RE-R9 ALSO USED BY SPDEV-ROUTINE
(13C)	316	SIGNED	4	CPWMRS (12)	REGISTER SAVE AREA FOR MESSAGE SUBROUTINE RE-R9
GENERAL WORK-AREA THE FOLLOWING SIXTY FOUR BYTES ARE USED AS GENERAL PURPOSE WORKAREA, WHICH MAY BE BROKEN DOWN INTO FIELDS AS IS REQUIRED					
(16C)	364	CHARACTER	32	CPWGW1	WORKAREA 1
(18C)	396	CHARACTER	32	CPWGW2	WORKAREA 2
MESSAGE OUTPUT AREA					
(1AC)	428	CHARACTER	30	MESSNMR	NODAL MESSAGE OUTPUT AREA
(1CA)	458	CHARACTER	132	MESSOUT (0)	MESSAGE OUTPUT/MODIFICATION AREA
(1CA)	458	BITSTRING	1	MESSLEN	LENGTH OF MESSAGE
(1CB)	459	CHARACTER	131	MESSAGE (0)	MESSAGE AREA
(1CB)	459	CHARACTER	7	MESSID	MESSAGE IDENTIFIER
(1D2)	466	CHARACTER	123	MESSTXT	MESSAGE TEXT
(24D)	589	CHARACTER	1		USED FOR ALIGNMENT
FIXED FORMAT OPERAND AREA					
(24D)	589		0	OPAREA	"*" BEGIN OF FIXED FORMAT OPERAND AREA
(24E)	590	CHARACTER	34	OP1 (0)	OPERAND 1
(24E)	590	BITSTRING	1	OPLEN1	LENGTH OF OPERAND CONTENTS
(24F)	591	BITSTRING	1	OPSW1	FLAG BYTE
(250)	592	BITSTRING	1		FLAG BYTE 2
(251)	593	BITSTRING	1		MASK BYTE
(252)	594	CHARACTER	24		OPERAND
(26A)	618	BITSTRING	2	OP1HEX	HEXADECIMAL VALUE OF OPERAND
(26C)	620	BITSTRING	4	OP1DEC	DECIMAL VALUE OF OPERAND
(270)	624	CHARACTER	34	OP2 (0)	OPERAND 2
(270)	624	BITSTRING	1	OPLEN2	LENGTH OF OPERAND CONTENTS
(271)	625	BITSTRING	1	OPSW2	FLAG BYTE
(272)	626	BITSTRING	1		FLAG BYTE 2
(273)	627	BITSTRING	1		MASK BYTE
(274)	628	CHARACTER	24		OPERAND
(28C)	652	BITSTRING	2	OP2HEX	HEXADECIMAL VALUE OF OPERAND
(28E)	654	BITSTRING	4	OP2DEC	DECIMAL VALUE OF OPERAND
(292)	658	CHARACTER	34	OP3 (0)	OPERAND 3
(292)	658	BITSTRING	1	OPLEN3	LENGTH OF OPERAND CONTENTS
(293)	659	BITSTRING	1	OPSW3	FLAG BYTE
(294)	660	BITSTRING	1		FLAG BYTE 2
(295)	661	BITSTRING	1		MASK BYTE

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(296)	662	CHARACTER	24		OPERAND
(2AE)	686	BITSTRING	2	OP3HEX	HEXADECIMAL VALUE OF OPERAND
(2B0)	688	BITSTRING	4	OP3DEC	DECIMAL VALUE OF OPERAND
(2B4)	692	CHARACTER	34	OP4 (0)	OPERAND 4
(2B4)	692	BITSTRING	1	OPLEN4	LENGTH OF OPERAND CONTENTS
(2B5)	693	BITSTRING	1	OPSW4	FLAG BYTE
(2B6)	694	BITSTRING	1		FLAG BYTE 2
(2B7)	695	BITSTRING	1		MASK BYTE
(2B8)	696	CHARACTER	24		OPERAND
(2D0)	720	BITSTRING	2	OP4HEX	HEXADECIMAL VALUE OF OPERAND
(2D2)	722	BITSTRING	4	OP4DEC	DECIMAL VALUE OF OPERAND
(2D6)	726	CHARACTER	34	OP5 (0)	OPERAND 5
(2D6)	726	BITSTRING	1	OPLEN5	LENGTH OF OPERAND CONTENTS
(2D7)	727	BITSTRING	1	OPSW5	FLAG BYTE
(2D8)	728	BITSTRING	1		FLAG BYTE 2
(2D9)	729	BITSTRING	1		MASK BYTE
(2DA)	730	CHARACTER	24		OPERAND
(2F2)	754	BITSTRING	2	OP5HEX	HEXADECIMAL VALUE OF OPERAND
(2F4)	756	BITSTRING	4	OP5DEC	DECIMAL VALUE OF OPERAND
(2F8)	760	CHARACTER	34	OP6 (0)	OPERAND 6
(2F8)	760	BITSTRING	1	OPLEN6	LENGTH OF OPERAND CONTENTS
(2F9)	761	BITSTRING	1	OPSW6	FLAG BYTE
(2FA)	762	BITSTRING	1		FLAG BYTE 2
(2FB)	763	BITSTRING	1		MASK BYTE
(2FC)	764	CHARACTER	24		OPERAND
(314)	788	BITSTRING	2	OP6HEX	HEXADECIMAL VALUE OF OPERAND
(316)	790	BITSTRING	4	OP6DEC	DECIMAL VALUE OF OPERAND
(31A)	794	CHARACTER	34	OP7 (0)	OPERAND 7
(31A)	794	BITSTRING	1	OPLEN7	LENGTH OF OPERAND CONTENTS
(31B)	795	BITSTRING	1	OPSW7	FLAG BYTE
(31C)	796	BITSTRING	1		FLAG BYTE 2
(31D)	797	BITSTRING	1		MASK BYTE
(31E)	798	CHARACTER	24		OPERAND
(336)	822	BITSTRING	2	OP7HEX	HEXADECIMAL VALUE OF OPERAND
(338)	824	BITSTRING	4	OP7DEC	DECIMAL VALUE OF OPERAND
(33C)	828	CHARACTER	34	OP8 (0)	OPERAND 8
(33C)	828	BITSTRING	1	OPLEN8	LENGTH OF OPERAND CONTENTS
(33D)	829	BITSTRING	1	OPSW8	FLAG BYTE
(33E)	830	BITSTRING	1		FLAG BYTE 2
(33F)	831	BITSTRING	1		MASK BYTE
(340)	832	CHARACTER	24		OPERAND
(358)	856	BITSTRING	2	OP8HEX	HEXADECIMAL VALUE OF OPERAND
(35A)	858	BITSTRING	4	OP8DEC	DECIMAL VALUE OF OPERAND
(35E)	862	CHARACTER	34	OP9 (0)	OPERAND 9
(35E)	862	BITSTRING	1	OPLEN9	LENGTH OF OPERAND CONTENTS
(35F)	863	BITSTRING	1	OPSW9	FLAG BYTE
(360)	864	BITSTRING	1		FLAG BYTE 2
(361)	865	BITSTRING	1		MASK BYTE
(362)	866	CHARACTER	24		OPERAND
(37A)	890	BITSTRING	2	OP9HEX	HEXADECIMAL VALUE OF OPERAND
(37C)	892	BITSTRING	4	OP9DEC	DECIMAL VALUE OF OPERAND

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(380)	896	CHARACTER	34	OP10 (0)	OPERAND 10
(380)	896	BITSTRING	1	OPLEN10	LENGTH OF OPERAND CONTENTS
(381)	897	BITSTRING	1	OPSW10	FLAG BYTE
(382)	898	BITSTRING	1		FLAG BYTE 2
(383)	899	BITSTRING	1		MASK BYTE
(384)	900	CHARACTER	24		OPERAND
(39C)	924	BITSTRING	2	OP10HEX	HEXADECIMAL VALUE OF OPERAND
(39E)	926	BITSTRING	4	OP10DEC	DECIMAL VALUE OF OPERAND
(3A2)	930	CHARACTER	34	OP11 (0)	OPERAND 11
(3A2)	930	BITSTRING	1	OPLEN11	LENGTH OF OPERAND CONTENTS
(3A3)	931	BITSTRING	1	OPSW11	FLAG BYTE
(3A4)	932	BITSTRING	1		FLAG BYTE 2
(3A5)	933	BITSTRING	1		MASK BYTE
(3A6)	934	CHARACTER	24		OPERAND
(3BE)	958	BITSTRING	2	OP11HEX	HEXADECIMAL VALUE OF OPERAND
(3C0)	960	BITSTRING	4	OP11DEC	DECIMAL VALUE OF OPERAND
(3C4)	964	CHARACTER	34	OP12 (0)	OPERAND 12
(3C4)	964	BITSTRING	1	OPLEN12	LENGTH OF OPERAND CONTENTS
(3C5)	965	BITSTRING	1	OPSW12	FLAG BYTE
(3C6)	966	BITSTRING	1		FLAG BYTE 2
(3C7)	967	BITSTRING	1		MASK BYTE
(3C8)	968	CHARACTER	24		OPERAND
(3E0)	992	BITSTRING	2	OP12HEX	HEXADECIMAL VALUE OF OPERAND
(3E2)	994	BITSTRING	4	OP12DEC	DECIMAL VALUE OF OPERAND
(3E6)	998	CHARACTER	34	OP13 (0)	OPERAND 13
(3E6)	998	BITSTRING	1	OPLEN13	LENGTH OF OPERAND CONTENTS
(3E7)	999	BITSTRING	1	OPSW13	FLAG BYTE
(3E8)	1000	BITSTRING	1		FLAG BYTE 2
(3E9)	1001	BITSTRING	1		MASK BYTE
(3EA)	1002	CHARACTER	24		OPERAND
(402)	1026	BITSTRING	2	OP13HEX	HEXADECIMAL VALUE OF OPERAND
(404)	1028	BITSTRING	4	OP13DEC	DECIMAL VALUE OF OPERAND
(408)	1032	CHARACTER	34	OP14 (0)	OPERAND 14
(408)	1032	BITSTRING	1	OPLEN14	LENGTH OF OPERAND CONTENTS
(409)	1033	BITSTRING	1	OPSW14	FLAG BYTE
(40A)	1034	BITSTRING	1		FLAG BYTE 2
(40B)	1035	BITSTRING	1		MASK BYTE
(40C)	1036	CHARACTER	24		OPERAND
(424)	1060	BITSTRING	2	OP14HEX	HEXADECIMAL VALUE OF OPERAND
(426)	1062	BITSTRING	4	OP14DEC	DECIMAL VALUE OF OPERAND
(42A)	1066	CHARACTER	34	OP15 (0)	OPERAND 15
(42A)	1066	BITSTRING	1	OPLEN15	LENGTH OF OPERAND CONTENTS
(42B)	1067	BITSTRING	1	OPSW15	FLAG BYTE
(42C)	1068	BITSTRING	1		FLAG BYTE 2
(42D)	1069	BITSTRING	1		MASK BYTE
(42E)	1070	CHARACTER	24		OPERAND
(446)	1094	BITSTRING	2	OP15HEX	HEXADECIMAL VALUE OF OPERAND
(448)	1096	BITSTRING	4	OP15DEC	DECIMAL VALUE OF OPERAND
		 1111		OPARNO	"(*-OPAREA)/(OP2-OP1)" MAX. NUMBER OF OPERANDS

DUMMY LAST OPERAND TO PREVENT ROUTINES FROM CHECKING BEYOND THE ABOVE LAST VALID OPERAND.

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(44C)	1100	CHARACTER	34	(0)	DUMMY LAST OPERAND
(44C)	1100	BITSTRING	1		LENGTH OF OPERAND CONTENTS
(44D)	1101	BITSTRING	1		FLAG BYTE
(44E)	1102	BITSTRING	1		FLAG BYTE 2
(44F)	1103	BITSTRING	1		MASK BYTE
(450)	1104	CHARACTER	24		OPERAND
(468)	1128	BITSTRING	2		HEXADECIMAL VALUE OF OPERAND
(46A)	1130	BITSTRING	4		DECIMAL VALUE OF OPERAND
(46A)	1130		0	OPNEXT	"*" END OF FORMAT OPERAND AREA
(46A)	1130		0	OPARLN	"*-OPAREA" LENGTH OF FORMATTED OPERAND AREA
	DUMMY TCB DEFINITION				
(470)	1136	DBL WORD	8	(0)	FORCE DOUBLE WORD ALIGNMENT
	THE FOLLOWING FIELDS DEFINE THE IDENTITY OF THE TASK, ESTABLISH ITS POSITION IN THE TASK LIST, RECORD PAGE FAULTS PENDING, AND DEFINE THE TASK STATE AT ANY POINT IN TIME.				
(470)	1136	CHARACTER	16	DYSD (0)	STORAGE DESCRIPTOR
(470)	1136	CHARACTER	4	DYBI	BLOCK IDENTIFIER
(474)	1140	CHARACTER	4	DYTI	TASK IDENTIFIER C'O CP' - COMMAND PROCESSOR TASK C'I IT' - INITIATOR TASK C'T TT' - TERMINATOR TASK C'T TI' - TIMER TASK C'RRDR' - LOCAL RDR TASK OR OFFLOAD C'WLST' - LOCAL PRT TASK OR OFFLOAD C'WPUN' - LOCAL PUN TASK C'E XX' - EXECUTION PROCESSOR TASK. XX SPECIFIES THE PARTITION REQUESTING THE TASK. C'1'-C'5 ' TCB BELONGS TO RJE TASK IN THIS CASE THREE REMAINING BYTES WILL INDICATE THE TYPE OF TASK. (RDR, LST, PUN, LGN, LGF, OR MSG.) C'LRLM' - LINE MANAGER TASK C'P PS' - PRINT STATUS TASK C' ACT' - ACCOUNT TASK C'J ' - SPOOL MANAGER TASK. THE THREE REMAINING BYTES IND THE TYPE OF TASK.(RDR,LST,OR SPM.) C'LSNA' - SNA TASK C'NTFY' - NOTIFY TASK C'LLDR' - PNET DRIVER C'NRVN' - NETWORK RECEIVER TASK N (N=BLANK FOR CONSOLE TASK) C'NTRN' - NETWORK TRANSMITTER N (N=BLANK FOR CONSOLE TASK) C'NCT ' - PNET SESSION EST'D C'NDT ' - PNET SESSION DISCONNECT C'XMAS' - SAS MASTER TASK C'XSAS' - SAS USER TASK C'XDEV' - DEVICE SERVICE TASK C'YTES' - TIME EVENT SCHEDULING C'DPST' - DYNAMIC PART.SCHEDULING
(478)	1144	CHARACTER	4	DYCU	PHYSICAL DEVICE IDENTIFIER
(47C)	1148	CHARACTER	4	DYRI (0)	RJE-ID/TAPE CUU/RCV-TSM TYPE
(47C)	1148	BITSTRING	1	DYFL	BINARY FORMAT
(47D)	1149	CHARACTER	3	DYRM	CHARACTER FORMAT
(480)	1152	ADDRESS	4	DYTP	ADDRESS OF PREVIOUS TASK TCB
(484)	1156	ADDRESS	4	DYTN	ADDRESS OF NEXT TASK TCB
(488)	1160	SIGNED	4	DYPF	PAGE FAULT REQUEST WORD
(48C)	1164	SIGNED	4	DYSF (0)	TASK SELECTION FIELD
(48C)	1164	BITSTRING	1		.. TASK STATE (SEE BELOW)
(48D)	1165	ADDRESS	3		.. NUCLEUS TASK ROUT. ADDR
(490)	1168	SIGNED	4	DYCT (4)	TASK CLASS LIST
		.... .1..		DY#C	"(*-DYCT)/4" .. NUMBER OF CLASS ENTRIES
(4A0)	1184	SIGNED	4	(0)	THE FOLLOWING TWO BYTES MUST BEGIN FULL WORD BOUNDARY |
(4A0)	1184	BITSTRING	1	DYFF	LIST DELIMITER | |

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
				|	
	TERMINATION TYPE |				
				|	
				|	
(4A1)	1185	BITSTRING	1	DYTT	TERMINATION TYPE (BELOW) V
(4A2)	1186	BITSTRING	1	DYJB	JOB BOUNDARY SWITCH
EQU X'FF' ..JOB IN PROCESS OVERLAY AREA USED BY X-PARTITION SPOOL MANAGER TASKS					
(494)	1172	BITSTRING	1		SPOOL MGMT LIST DELIM'TER
(495)	1173	ADDRESS	3	DYEWA	ADDR. OF WS FOR EXTRACT
(498)	1176	BITSTRING	1	DYIQ	SPOOL MGMT QUEUE ID
(499)	1177	BITSTRING	1		UNUSED
(49A)	1178	BITSTRING	1	DYSG	SPOOL MG GEN PURPOSE BYTE
		.... ..1.		DY1T	"X'02'" .. 1ST TIME BUFF'ED GETSP
EQU X'01' .. PUTSPOOL DASD SOS MSG					
(49B)	1179	CHAR-ACTER	1	DYSS	SPOOL MANAGEMENT SWITCH
		11.. 1..1		DYIW	"C'I'" ..LOGICAL WRITER INITIALIZED
		11.1 .11.		DYOW	"C'O'" ..OPEN LOGICAL WRITER
		11.. ..11		DYCW	"C'C'" ..CLOSE LOGICAL WRITER
(49C)	1180	SIGNED	4	DYER	ADDR(USER X-PART XECB)
	FUNCTION TRACE INDICATOR				
(4A3)	1187	BITSTRING	1	DYFT	FUNCTION TRACE INDICATOR
	TASK ECB AND OTHER CONTROL FLAGS				
(4A4)	1188	SIGNED	4	DYEB (0)	EVENT CONTROL BLOCK
(4A4)	1188	BITSTRING	1	DYDB	DOUBLE BUFFER INDICATOR
(4A5)	1189	BITSTRING	1	DYCB	FUNCTION COMMUNICATION BYTE
(4A6)	1190	BITSTRING	1	DYEP	EVENT POST BYTE
		1...		DYEO	"X'80'" EVENT POST BIT ON SETTING
		.1..		DYBSCLV	"X'40'" EVENT BIT BSC-WAIT 'B'
		..1.		DYQRDR	"X'20'" POST BIT FOR QUIESCE RDR I/O
(4A7)	1191	BITSTRING	1	DYSI	SPOOLING INDICATOR
(4A8)	1192	ADDRESS	4		UNUSED
(4AC)	1196	ADDRESS	4		UNUSED
	TASK REGISTER SAVE AREA THE FOLLOWING FIELDS RECORD THE CONTENTS OF THE GENERAL PURPOSE REGISTERS 12 THROUGH 9 WHENEVER ENTRY IS MADE TO TASK SELECTION. IF THE TASK STATE IS SET TO 'R' (RUNNING) THE VALUES IN THE FIELDS RECORD THE CONTENTS OF THE REGISTERS WHEN THE TASK WAS GIVEN CONTROL. IF THE TASK STATE IS SET TO ANY OTHER VALUE THE FIELDS CONTAIN THE ACTUAL CONTENTS OF THE REGISTERS ASSOCIATED WITH THE TASK.				
(4B0)	1200	CHAR-ACTER	56	DYTR (0)	TASK REGISTER SAVE AREA
(4B0)	1200	SIGNED	4	DYRC	TASK REGISTER 12
(4B4)	1204	SIGNED	4	DYRD	TASK REGISTER 13
(4B8)	1208	SIGNED	4	DYRE	TASK REGISTER 14
(4BC)	1212	SIGNED	4	DYRF	TASK REGISTER 15
(4C0)	1216	SIGNED	4	DYR0	TASK REGISTER 0
(4C4)	1220	SIGNED	4	DYR1	TASK REGISTER 1
(4C8)	1224	SIGNED	4	DYR2	TASK REGISTER 2
(4CC)	1228	SIGNED	4	DYR3	TASK REGISTER 3
(4D0)	1232	SIGNED	4	DYR4	TASK REGISTER 4
(4D4)	1236	SIGNED	4	DYR5	TASK REGISTER 5
(4D8)	1240	SIGNED	4	DYR6	TASK REGISTER 6
(4DC)	1244	SIGNED	4	DYR7	TASK REGISTER 7
(4E0)	1248	SIGNED	4	DYR8	TASK REGISTER 8
(4E4)	1252	SIGNED	4	DYR9	TASK REGISTER 9

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
	VARIOUS CONTROL FIELDS				
(4E8)	1256	SIGNED	4	DYRS (0)	RESTART INFORMATION
(4E8)	1256	SIGNED	4	(0)	TASK TERMINATOR WORK AREA
(4E8)	1256	SIGNED	4	(0)	IPW$$XTC ECB FOR DISPLAY SPOOL LST
(4E8)	1256	BITSTRING	1	DYRX	RESTART FUNCTION INDEX

EQU X'04' ..RESTART REQUESTED,NO SIGN
EQU X'08' ..RESTART REQUESTED, + SIGN
EQU X'0C' ..RESTART REQUESTED, - SIGN

		...1		DYSP	"X'10'" ..SETUP REQUESTED
		...1 1...		DYCKP	"X'18'" ..CHECKPOINT REQUEST
		...1 11..		DYPAE	"X'1C'" ..POSITION AT END IF ERROR
(4E9)	1257	ADDRESS	3		RESERVED FOR FUTURE USE
(4E9)	1257	BITSTRING	1	DYCTRC	CURRENT TRC COMMAND CODE
(4EA)	1258	SIGNED	2	DYBL	BUFFER LENGTH
(4E8)	1256	BITSTRING	1	DYRYFRB	FUNCT. REQ. BYTE OF CALLER
(4E9)	1257	BITSTRING	1	DYRYTD	HELP FIELD USED BY RECOVERY
(4EC)	1260	BITSTRING	1	DYDT	DEVICE TYPE CODE
(4ED)	1261	BITSTRING	1	DYAT	ACCOUNT TRACE INDICATOR
(4EE)	1262	BITSTRING	2	DYDE	PACKED DEVICE ADDRESS
(4F0)	1264	SIGNED	4	DYRG	SAVE AREA FOR SERV. RTNS
(4F4)	1268	SIGNED	4	DYRH	SAVE AREA FOR SERV RTNS
(4F8)	1272	SIGNED	4	DY15	2ND BASE REG. SAVE AREA

WHENEVER A VSE/POWER SERVICE FUNCTION IS CALLED (EXCEPT TASK MANAGEMENT) REGISTER 9 IS SAVED IN TC09. REGISTER 9 IS THEN USED AS 2ND BASE REGISTER BY THE NUCLEUS ROUTINES. REGISTER 8 IS SAVED IN TC08 TO BE USED AS 3RD BASE.

(4FC)	1276	SIGNED	4	DY08	REGISTER 8 SAVE AREA
(500)	1280	SIGNED	4	DY09	REGISTER 9 SAVE AREA

LINKAGE REGISTER SAVE AREA
THE FOLLOWING FIELDS RECORD THE CONTENTS OF THE GENERAL PURPOSE REGISTERS 14 THROUGH 9 WHENEVER ENTRY IS MADE BY THE TASK TO A VSE/POWER FUNCTION.

(504)	1284	CHARACTER	56	DYSV (0)	REGISTER SAVE AREA
(504)	1284	SIGNED	4		TASK CONTROL ADDRESS
(508)	1288	SIGNED	4		PREVIOUS SAVE AREA ADDRESS
(50C)	1292	SIGNED	4		SAVED REGISTER 14
(510)	1296	SIGNED	4		SAVED REGISTER 15
(514)	1300	SIGNED	4		SAVED REGISTER 0
(518)	1304	SIGNED	4		SAVED REGISTER 1
(51C)	1308	SIGNED	4		SAVED REGISTER 2
(520)	1312	SIGNED	4		SAVED REGISTER 3
(524)	1316	SIGNED	4		SAVED REGISTER 4
(528)	1320	SIGNED	4		SAVED REGISTER 5
(52C)	1324	SIGNED	4		SAVED REGISTER 6
(530)	1328	SIGNED	4		SAVED REGISTER 7
(534)	1332	SIGNED	4		SAVED REGISTER 8
(538)	1336	SIGNED	4		SAVED REGISTER 9

TRACE FACILITY SAVE AREA

(53C)	1340	SIGNED	4	DYTCWKP	TASK TRACE WORKAREA PNTR
(540)	1344	CHARACTER	0	DYTCR (0)	TASK TRACE REG SAVEAREA
(540)	1344	SIGNED	4	DYTCRD	TASK TRACE REG 13
(544)	1348	SIGNED	4	DYTCRE	TASK TRACE REG 14
(548)	1352	SIGNED	4	DYTCRF	TASK TRACE REG 15
(54C)	1356	SIGNED	4	DYTCR0	TASK TRACE REG 0
(550)	1360	SIGNED	4	DYTCR1	TASK TRACE REG 1
(554)	1364	SIGNED	4	DYTCR2	TASK TRACE REG 2
(558)	1368	SIGNED	4	DYTCR3	TASK TRACE REG 3
(55C)	1372	SIGNED	4	DYTCR4	TASK TRACE REG 4

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(560)	1376	SIGNED	4	DYTCR5	TASK TRACE REG 5
(564)	1380	SIGNED	4	DYTCR6	TASK TRACE REG 6
(568)	1384	SIGNED	4	DYTCR7	TASK TRACE REG 7
(56C)	1388	SIGNED	4	DYTCR8	TASK TRACE REG 8
(570)	1392	SIGNED	4	DYTCR9	TASK TRACE REG 9

IDUMP SAVE AREA
WHENEVER THE IDUMP FUNCTION IS REQUESTED BY IPW$IDM,
CALLER REGISTERS RE-R1 ARE SAVED IN TCIE-TCI1
NOTE: THIS AREA IS REUSED AS ECB LIST FOR IPW$WFM CALLS BY
IPW$$QM(Q1), IPW$$T1, IPW$$XWE.
IN $WFM STATE IDUMP IS NOT POSSIBLE EXCEPT DISPATCHER
DETECTS DESTROYED TCB AND TERMINATES POWER.

(574)	1396	SIGNED	4	DYIE	REGISTER E SAVE AREA
(578)	1400	SIGNED	4	DYIF	REGISTER F SAVE AREA
(57C)	1404	SIGNED	4	DYI0	REGISTER 0 SAVE AREA
(580)	1408	SIGNED	4	DYI1	REGISTER 1 SAVE AREA

TASK MESSAGE INTERFACE

(584)	1412	ADDRESS	4	DYMW	ADDRESS OF MESSAGE TO BE ISSUED

EQU X'80' .. HOLD MESSAGE CONTROL BLOCK
EQU X'40' .. REGISTER 5 CONTAINS TCB ADDRESS
EQU X'20' .. MESSAGE IS IN NMR FORMAT

(588)	1416	ADDRESS	4	DYAW	ADDRESS OF CALLERS REPLY AREA

WTO/WTOR/DOM INTERFACE
WTO/WTOR OUTPUT

(58C)	1420	BITSTRING	4	DYMID	MESSAGE ID
(590)	1424	ADDRESS	4	DYMRECB	WTOR REPLY ECB

WTO/WTOR/DOM INPUT - SET BY POWER:

(594)	1428	BITSTRING	4	DYMRT	MESSAGE ROUTING CODE
(598)	1432	BITSTRING	2	DYMDC	MESSAGE DESCRIPTOR CODE
(59A)	1434	BITSTRING	1	DYF18	FLAG BYTE 18
(59B)	1435	BITSTRING	1	DYF19	FLAG BYTE 19
(59C)	1436	BITSTRING	4	DYMNRT	NEG ROUTING CODE(DON"T WANT)@D61CDSW
(5A0)	1440	BITSTRING	4	DYMRTDF	DEFAULT MSG ROUTING CODE
(5A4)	1444	ADDRESS	4	DYMDOM	DOM MESSAGE ID
(5A8)	1448	ADDRESS	4	DYMCID	COMMAND CONNECT MESSAGE ID

AR COMMAND OUTPUT(INPUT TO WTO/WTOR IF CMD RESP)

(5AC)	1452	BITSTRING	8	DYMCART	AR MESSAGE TOKEN (CART)
(5B4)	1460	BITSTRING	4	DYMCOID	AR CONSOLE ID

MISCELLANEOUS

(5B8)	1464	BITSTRING	2	DYMPID	PARTITION ID
(5BA)	1466	BITSTRING	1	DYMFLG	MESSAGE FLAGS
		1...		DYMFAR	"X'80'" .. VSE/AF CMD
		.1..		DYMFUR	"X'40'" .. VSE/AF CMD USER CONSOLE
		..1.		DYMFCUP	"X'20'" .. CLOSE UP CONN'D MSGS
		...1		DYMFCFM	"X'10'" .. ISSUE 1ST CONN'D MESSAGE
		.... 1...		DYMFICM	"X'08'" .. ISSUE CONNECTED MESSAGE
		.... .1..		DYMFCEX	"X'04'" .. CONNECTED MSG EXISTS
(5BB)	1467	BITSTRING	1		UNUSED
(5BC)	1468	ADDRESS	4	DYVD	SAVED PTR(LINK-REG-SV-AREA)
(5C0)	1472	ADDRESS	4	DY1Q40	MSG 1Q40A MSG ID FOR DOM
(5C0)	1472	ADDRESS	4	DY1Q38	MSG 1Q38A MSG ID FOR DOM
(5C0)	1472	ADDRESS	4	DY1QD6	MSG 1QD6I MSG ID FOR DOM
(5C0)	1472	ADDRESS	4	DY1QE6	MSG 1QE6I MSG ID FOR DOM
(5C0)	1472	ADDRESS	4	DY1QD7	MSG 1QD7I MSG ID FOR DOM
(5C4)	1476	ADDRESS	4		UNUSED
(5C8)	1480	ADDRESS	4	DYPFTWA	TAPE-WORKAREA NOT TO REL.
(5CC)	1484	ADDRESS	4	DYPFPWA	PRT/PUN-WA NOT TO RELEASE
(5D0)	1488	SIGNED	2	DYPFCU	CUU OF PSTOP FORCE CMD

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(5D2)	1490	SIGNED	2	DYPFRC	RC OF PSTOP FORCE CMD
		DATA FILE CONTROL WORDS INPUT/OUTPUT REQUEST WORD			
(5D4)	1492	ADDRESS	4	DYDW	RELATIVE DBLK NUMBER
(5D8)	1496	ADDRESS	4	DYDV	VIRTUAL DATA AREA ADDRESS
(5DC)	1500	ADDRESS	2	DYDL	DATA AREA LENGTH
(5DE)	1502	BITSTRING	1		OPERATION CODE
(5DF)	1503	BITSTRING	1		RESERVED
		BLOCKING CONTROL WORDS			
(5E0)	1504	SIGNED	4	DYBC	RESIDUAL BLOCK COUNT
(5E4)	1508	ADDRESS	4	DYPR	PREVIOUS/NEXT RECORD ADDRESS
(5E8)	1512	ADDRESS	4	DYASPB	ADDR OF SPOOL BLOCK
		RECORD CONTROL WORD (RCW)			
(5EC)	1516	CHARACTER	8	DYRW (0)	RECORD CONTROL WORD
(5EC)	1516	BITSTRING	1	DYCC	RECORD COMMAND CODE X'FF' .. CONTROL RECORD X'FE' .. NEW FORMS
(5ED)	1517	ADDRESS	3	DYRV	RECORD ADDRESS (VIRTUAL)
(5F0)	1520	BITSTRING	1	DYGP	GENERAL PURPOSE BYTE
(5F1)	1521	BITSTRING	1	DYG2	GENERAL PURPOSE BYTE TWO
(5F2)	1522	SIGNED	2	DYRL	RECORD LENGTH
(5F4)	1524	BITSTRING	1	DYG3	GENERAL PURPOSE BYTE THREE
(5F5)	1525	BITSTRING	1	DYG4	GENERAL PURPOSE BYTE FOUR
(5F6)	1526	BITSTRING	1	DYDVEB	'DEVICE END' OCCURRED ('FF')
(5F7)	1527	BITSTRING	1		RESERVED FOR FUTURE USE
(5F8)	1528	SIGNED	4	DYLRNO	LOGICAL RECORD NUMBER
		QUEUE FILE CONTROL WORDS			
(5FC)	1532	ADDRESS	4	DYQW	RELATIVE QUEUE REC NUMBER
(600)	1536	ADDRESS	4	DYQV	VIRTUAL SPACE ADDRESS
(604)	1540	ADDRESS	2		QUEUE REC LENGTH - NOT USED
(606)	1542	BITSTRING	1		OPERATION CODE
(607)	1543	BITSTRING	1		RESERVED
		TAPE SPOOLING CONTROL INFORMATION			
(608)	1544	BITSTRING	8	DYTS (0)	TAPE REQUEST WORD
(608)	1544	BITSTRING	1	DYTF	FUNCTION BYTE USED FOR TAPE
(609)	1545	ADDRESS	3	DYTA	ADDRESS OF TAPE CTRL BLOCK
(60C)	1548	BITSTRING	4	DYTDES (0)	TAPE UNIT DESCRIPTORS
(60C)	1548	BITSTRING	1	DYTM	INDICATE TAPE MODE (DENS.)
(60D)	1549	BITSTRING	1	DYTDT	TAPE DEVICE TYPE
(60E)	1550	CHARACTER	2	DYTU	TAPE PROG.LOGICAL UNIT(PUU)
(610)	1552	SIGNED	4	DYPU	PHYS.UNIT OR PUB ENTRY ADDR.
		VARIOUS CONTROL FIELDS			
(614)	1556	BITSTRING	1	DYF2	FLAG BYTE 2
(615)	1557	BITSTRING	1	DYF3	FLAG BYTE 3
(616)	1558	BITSTRING	1	DYF4	FLAG BYTE 4
(617)	1559	BITSTRING	1	DYF5	FLAG BYTE 5
(618)	1560	BITSTRING	1	DYF6	FLAG BYTE 6
(619)	1561	BITSTRING	1	DYF7	FLAG BYTE 7
(61A)	1562	BITSTRING	1	DYF8	FLAG BYTE 8
(61B)	1563	BITSTRING	1	DYF9	FLAG BYTE 9
(61C)	1564	BITSTRING	1	DYF10	FLAG BYTE 10
(61D)	1565	BITSTRING	1	DYF11	FLAG BYTE 11
(61E)	1566	BITSTRING	1	DYF12	FLAG BYTE 12
(61F)	1567	BITSTRING	1	DYF13	FLAG BYTE 13 (RES.FOR GCM)
(620)	1568	BITSTRING	1	DYF14	FLAG BYTE 14
(621)	1569	BITSTRING	1	DYF15	FLAG BYTE 15
(622)	1570	BITSTRING	1	DYF16	FLAG BYTE 16
(623)	1571	BITSTRING	1	DYF17	FLAG BYTE 17

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(624)	1572	BITSTRING	1	DYF20	FLAG BYTE 20
(625)	1573	BITSTRING	1	DYF21	FLAG BYTE 21
(626)	1574	BITSTRING	1	DYF22	FLAG BYTE 22
(627)	1575	BITSTRING	1	DYF23	FLAG BYTE 23
(628)	1576	BITSTRING	4		.. UNUSED
(62C)	1580	SIGNED	4	DYVEB (0)	ECB FOR VTAM REQUESTS
ALSO USED BY $$CA TO INDICATE EXEC. WRITER TO SEGMENT					
		1...		DYVEBPG	"X'80'" .. SEGMENT PAGE REQ. EX.WTR
		.1..		DYVEBIM	"X'40'" .. SEGMENT IMM. REQ. EX.WTR
		..1.		DYVEBIF	"X'20'" .. IMM. SEGMENTATION FORCED
(62C)	1580	BITSTRING	2		RESERVED - DO NOT USE !
(62E)	1582	BITSTRING	1	DYVEP	EVENT POST BYTE FOR VTAM
		1...		DYVEO	"X'80'" ..EVENT POST BIT ON
(62F)	1583	BITSTRING	1		RESERVED - DO NOT USE !
(62C)	1580	SIGNED	4	DYFEB (0)	ECB FOR FORMAT DATA FILE
AREA REUSED TO POST REQUEST FROM $$I4/$$RY TO $$T1					
(62C)	1580	BITSTRING	2		RESERVED - DO NOT USE !
(62E)	1582	BITSTRING	1	DYFEP	EVENT POST BYTE FOR VTAM
		1...		DYFEO	"X'80'" ..EVENT POST BIT ON
(62F)	1583	BITSTRING	1		RESERVED - DO NOT USE !
(630)	1584	ADDRESS	4	DYGDS	ADDR OF GENERATED DSHR
(634)	1588	ADDRESS	4	DYCMRG	COMREG ADDR IF EXEC TASK
(634)	1588	SIGNED	4	DYDEB	'CLEAN DELETION QUEUE' ECB
USED TO POST INIT/TERMIN. TASK FROM $$FQ AND $$PS					
(638)	1592	SIGNED	4	DY3E	ADDR OF TCB EXTENSION AREA OR ADDR OF WORK SPACE OR DSHR
(63C)	1596	SIGNED	4	DYHD	HEAD PTR VIRT STORAGE CHAIN
(640)	1600	SIGNED	4		TAIL PTR VIRT STORAGE CHAIN
(644)	1604	ADDRESS	4	DY3W	POINTER 3540 WORKSPACE
(648)	1608	SIGNED	4	DYXWA	ADDRESS TO EXIT WORK AREA
(64C)	1612	SIGNED	4	DYJHR	PTR TO JHR (USED BY $LR)
(650)	1616	SIGNED	2	DYXWAL	LENGTH OF EXIT WORK AREA
(652)	1618	SIGNED	2	DYQCQW	QUEUE REC. OF QC.. USED BY $SQ IN CASE OF SOD FOR QCA
(654)	1620	SIGNED	4	DYLRWA	LOGICAL READER WORK AREA
(658)	1624	SIGNED	4	DYRVAL	RESTART PAGE/LINE/RECORD CNT
(65C)	1628	SIGNED	4	DY0EEX	RETURN ADDRESS OF USER EXIT
(660)	1632	SIGNED	4	DYNR2W	PTR TO NR2 WORKAREA
LOGICAL I/F AND FUNCTION REQUEST BYTES					
(664)	1636	BITSTRING	1	DYLIFB	LOG. INTERFACE FUNCTION BYTE
		1		DYLIOP	"X'01'" .. PRE-OPEN OUTPUT QUEUE
		.... ..1.		DYLIOF	"X'02'" .. FINAL OPEN OUTPUT QUEUE
		11		DYLILO	"X'03'" .. LOCATE QUEUE ENTRY
		.... .1..		DYLIOR	"X'04'" .. OPEN-RESTART QUEUE ENTRY
		.... .1.1		DYLIRS	"X'05'" .. RESTART QUEUE ENTRY
		.... .11.		DYLISG	"X'06'" .. SEGMENT OUTPUT Q' ENTRY
		.... .111		DYLIFL	"X'07'" .. FLUSH OUTPUT QUEUE ENTRY
		.... 1...		DYLICH	"X'08'" .. CHECKPOINT QUEUE ENTRY
		.... 1..1		DYLICL	"X'09'" .. CLOSE OUTPUT QUEUE ENTRY
		.... 1.1.		DYLISP	"X'0A'" .. SPOOL RECORD
		.... 1.11		DYLIAQ	"X'0B'" .. ADD TO CLASS CHAIN
		.... 11..		DYLICO	"X'0C'" .. CLOSE WITHOUT JTR
		.... 11.1		DYLIRL	"X'0D'" .. READ LOCATED DATA
		.... 111.		DYLIOFJ	"X'0E'" .. FINAL OPEN OUTPUT QUEUE WITH JOBNUMBER SUPPLIED
		 1111		DYLIFLW	"X'0F'" .. FLUSH OUTPUT QUEUE ENTRY WITHOUT MESSAGE
		...1		DYLIOFB	"X'10'" .. FINAL OPEN OUTPUT QUEUE WITH JOBNUMBER SUPPLIED, NO BLANK TRUNCATION
(665)	1637	BITSTRING	1	DYLIRC	LOG. INTERFACE RETURN CODE
(666)	1638	BITSTRING	1	DYLISR	2ND LOG. INTERFACE RC

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
FUNCTION REQUEST BYTE 1 & 2 TRANSPORT INFORMATION TO THE CALLED FUNCTION, WHERE THEY ARE CLEARED. THIS ENABLES THE CALLED FUNCTION TO CALL ANOTHER FUNCTION AND USE THE FUNCTION BYTES FOR ITS PURPOSES.					
(667)	1639	BITSTRING	1	DYFRB1	FUNCTION REQUEST BYTE 1
(668)	1640	BITSTRING	1	DYFRCT	FUNCTION RETURN CODE
(669)	1641	BITSTRING	1	DYFRB2	FUNCTION REQUEST BYTE 2
CONTINUATION OF VARIOUS CONTROL FIELDS					
(66A)	1642	BITSTRING	1	DYRSCO	REMAINING COPY NUMBER
THE FOLLOWING FIELD IS USED BY THE EXECUTION PROCESSOR ROUTINES (IPW$$XR & IPW$$XW) TO INDICATE THE DETAIL REASON CODE FOR MESSAGE 1R30I.					
(66B)	1643	BITSTRING	1	DYLRRL	JOB REC LEN, 1ST SYSRDR DEV
(66C)	1644	SIGNED	4	DYPL	ADDR(SPOOL MGMT PARM LST)
EQU X'FF' .. SPOOL MGR SPL PRESENT					
(670)	1648	CHAR-ACTER	8	DYSECAU	SECURITY OWNING USERID
(678)	1656	BITSTRING	1	DYSECFG	SECURITY FLAGS
(679)	1657	BITSTRING	1	DYDMREP	TESTED BY IPW$$DS IN IPW$$DM TO REPLACE JOB OR DATA SET HEADERS, SET BY IPW$$I7 OR IPW$$OP
(67A)	1658	BITSTRING	2		UNUSED
(67C)	1660	SIGNED	4	DYDHD	HEAD PTR VIRT STORAGE CHAIN OF DUPL. Q-ENTRIS IN CREAT.
(680)	1664	SIGNED	4	DYDTL	TAIL PTR VIRT STORAGE CHAIN
(684)	1668	ADDRESS	4	(3)	UNUSED
GENERAL TASK WORK AREA THE FOLLOWING 32+16+16+16+72+64 = 216 BYTES ARE USED AS A GENERAL-PURPOSE WORK AREA, WHICH MAY BE BROKEN INTO FIELDS AS REQUIRED BY EACH SPECIFIC TASK.					
(690)	1680	SIGNED	4	(0)	
(690)	1680	BITSTRING	32	DYGW	WORK AREA - USED BY LOGICAL ROUTINES, MAY NOT BE REUSED BY ANY TASK USING LOG. RTNS
(6B0)	1712	SIGNED	4	DYW1	WORK WORD 1
(6B4)	1716	SIGNED	4	DYW2	WORK WORD 2
(6B8)	1720	SIGNED	4	DYW3	WORK FULLWORD 3
(6BC)	1724	SIGNED	4	DYW4	WORK FULLWORD 4
(6C0)	1728	SIGNED	4	(4)	RESERVED
(6D0)	1744	BITSTRING	16	DYGW2	WORK AREA 2
(6E0)	1760	BITSTRING	72	DYGW3	WORK AREA 3
(728)	1832	BITSTRING	64	DYGW4	WORK AREA 4
EXECUTION READER RE-DEFINITION					
(690)	1680	BITSTRING	8		USED FOR OTHER PURPOSES
(698)	1688	ADDRESS	4	DYXAAR	ADDRESS OF ACCOUNT RECORD
(69C)	1692	CHAR-ACTER	1	DYXACL	CLASS
(69D)	1693	BITSTRING	1	DYXFLG	FLAG BYTE
		1		DYXRDO	"X'01'" .. READ ONLY SWITCH
(69E)	1694	BITSTRING	2		UNUSED
(6A0)	1696	ADDRESS	4	DYXPDB	ADDRESS OF PART CNTL BLOCK
(6A4)	1700	BITSTRING	8	DYXJTD	JOB START TOD CLOCK
(6AC)	1708	ADDRESS	4	DYXWF0	EXEC. RDR. WORK FIELD 0
(6B0)	1712	ADDRESS	4	DYXWF1	EXEC. RDR. WORK FIELD 1
(6B4)	1716	ADDRESS	4	DYXWF2	EXEC. RDR. WORK FIELD 2
(6B8)	1720	ADDRESS	4	DYEBXR	XRE ECB DURING BAM OPEN
NOTE- THIS SECTION MAY NOT OVERWRITE TCGW2					
EXECUTION WRITER RE-DEFINITION					

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(690)	1680	BITSTRING	8		USED FOR OTHER PURPOSES
(698)	1688	ADDRESS	4	DYMTJH	MT PARTITION JHR POINTER
(69C)	1692	ADDRESS	4	DYMTJT	MT PARTITION JTR POINTER
(6A0)	1696	BITSTRING	16	DYLTAB	CARRIAGE CONTROL TABLE
(6B0)	1712	CHARACTER	8	DYSECU	VSE SECURITY USERID SAVEAREA
(6B8)	1720	CHARACTER	8	DYSECN	VSE SECURITY SECNODE " @KX40618
(6C0)	1728	ADDRESS	4	DYXWPDB	ADDRESS OF PART CNTL BLOCK

NOTE- THIS SECTION MAY NOT OVERWRITE TCGW2

EXECUTION READER AND WRITER RE-DEFINITION
NOTE: OVERLAYS WORK AREA 2 AND WORK AREA 3 AND WORK AREA 4

(6D0)	1744	SIGNED	4	DYXJS6	$$XJ SUBROUTINES BASE REG.
(6D4)	1748	SIGNED	4	DYJGM	EX. WRITER MESSAGE ADDR.
(6D8)	1752	SIGNED	4	DYPURC	EX. WRITER 'PURGE' RET-CODE
(6DC)	1756	ADDRESS	4	DYXRWA	EX. PROCESSOR WORK AREA
(6E0)	1760	SIGNED	4	DYALET	ALET FOR PARTITION
(6E4)	1764	BITSTRING	16	DYAAR (0)	SAVED ACC REG 1,6 - 8
(6E4)	1764	ADDRESS	4	DYAR1	SAVED ACC REG 1
(6E8)	1768	BITSTRING	12	DYARS (0)	SAVED ACC REG 6 - 8
(6E8)	1768	SIGNED	4	DYAR6	SAVED ACC REG 6
(6EC)	1772	SIGNED	4	DYAR7	SAVED ACC REG 7
(6F0)	1776	SIGNED	4	DYAR8	SAVED ACC REG 8
(6F4)	1780	SIGNED	4	DYAR2	SAVED ACC REG 2
(6F8)	1784	SIGNED	4	DYSVSP	TEMP STORAGE POINTER
(6FC)	1788	ADDRESS	4	DYXJNP	IPW$$XJ NEW TASK ADDR
(700)	1792	ADDRESS	2	DYSVSPL	LENGTH OF TCSVSP BUF
(702)	1794	ADDRESS	2	DYRRC	IPW$$XJ ERROR RETURN CODE
(704)	1796	SIGNED	4	DYSR4	SAVED REG.4 (EX.WRITER)
(708)	1800	ADDRESS	4	DYJGM2	COPY OF F.F. JOB GEN. MSG
(70C)	1804	SIGNED	4	DYQ25ID	MSG 1Q25A ID FROM IPW$$T1
(710)	1808	CHARACTER	7	DYXTLBL	IPW$$XJ BAM DTFNAME (LABEL)
(717)	1815	BITSTRING	1	DYXWFG	FLAGS
		1...		DYXFTLBL	"X'80'" .. IPW$$XJ TLBL= SPEC'D
		.1..		DYXFLTPY	"X'40'" .. IPW$$XJ LTAPE=YES SPEC'D
		..1.		DYXFLTPN	"X'20'" .. IPW$$XJ LTAPE=NO SPEC'D
		...1		DYXFIPC	"X'10'" .. IPW$$XWE SAVED PG-INCRM.
(718)	1816	SIGNED	4	DYXXJIW	ADDR OF IPW$$XJ INTERFACE WA
(71C)	1820	SIGNED	4	DYXXJSG (3)	SAVED REG. TEMP BY IPW$$XJ

NOTE- THIS SECTION MAY NOT EXTEND BEYOND TCGW4

WORK AREA FOR PLOAD COMMAND PROCESSOR

(6E0)	1760	CHARACTER	1	DYEXTY	EXIT TYPE
		11.1 ...1		DYEXJO	"C'J'" ..JOB EXIT
		11.1 .11.		DYEXOU	"C'O'" ..OUT EXIT
		11.1 .1.1		DYEXNE	"C'N'" ..NET EXIT
		111. .111		DYEXXM	"C'X'" ..XMT EXIT
(6E1)	1761	BITSTRING	3		RESERVED FOR FUTURE USE
(6E4)	1764	CHARACTER	8	DYEXNA	EXIT NAME
(6EC)	1772	SIGNED	4	DYEXSI	EXIT SIZE
(6F0)	1776	SIGNED	4	DYEXAD	EXIT LOAD POINT ADDRESS
(6F4)	1780	SIGNED	4	DYEXEP	EXIT ENTRY POINT ADDRESS
		...1 1...		DYEXLE	"*-DYEXTY" ..LENGTH OF WA

POFFLOAD TASK RE-DEFINITION

(6B0)	1712	CHARACTER	8	DYOONN	OLD NODE NAME
(6B8)	1720	BITSTRING	1	DYOFLG	FLAG BYTE
(6B9)	1721	BITSTRING	1	DYOSW1	SWITCH BYTE 1

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(6BA)	1722	BITSTRING	1	DYORC	SAVE AREA FOR RETURN CODE
(6BB)	1723	BITSTRING	1	DYOFL2	FLAG BYTE 2
(6BC)	1724	ADDRESS	4	DYOSAL	SELECT QUEUE ARGUMENT LIST
(6C0)	1728	CHARACTER	7	DYOTLBL	BAM DTF NAME (LABEL)
(6C7)	1735	CHARACTER	1	DYTBQI	POFFLOAD QUEUE ID 33
(6C8)	1736	SIGNED	2	DYOQRL	QRA LENGTH FOR PREVIOUS REL.
(6CA)	1738	BITSTRING	2		UNUSED
(6CC)	1740	SIGNED	4	DYPSHT	PICKUP TOT SCHEDULED ENTRIES
(6D0)	1744	SIGNED	4	DYPSAT	PICKUP TOT SAVED ENTRIES
(6D4)	1748	BITSTRING	8	DYPMSG	PICKUP MSG 1Q6PI TIMESTAMP
(6DC)	1756	BITSTRING	8	DYPTMP	PICKUP TEMP WORKAREA
(6E4)	1764	BITSTRING	17	DYPCLA	PICKUP SAVE OF TCCT
(6F5)	1781	BITSTRING	3		UNUSED
(6F8)	1784	ADDRESS	4	DYOFJCA	POFFLOAD JOURNALING JCA PNTR
(6FC)	1788	ADDRESS	4	DYOFJRD	POFFLOAD JOURNALING REG.13
(700)	1792	ADDRESS	4	DYOFJRD1	POFFLOAD JOURNALING REG.13
(704)	1796	CHARACTER	6	DYOCMDT	POFFLOAD CMD TYPE
(70A)	1802	SIGNED	2	DYOFTAP	POFFLOAD TAPE NO. NOW OFTAP=
(70C)	1804	ADDRESS	4	DYQRQN	NEXT QUEUE REC ADDR 20
(710)	1808	ADDRESS	4	DYOFNUM	TAPE ENTRY DEC SEQ NO.OFNUM=
(714)	1812	CHARACTER	4	DYO$OFJ	JOURNAL ID NNNN ($OFJNNNN)
(718)	1816	CHARACTER	8	DYOCRWK	CRDAYS WORKAREA
(720)	1824	CHARACTER	8	DYOCRDA	CRDAYS=
(728)	1832	BITSTRING	1	DYOCRMK	CRDAYS BRANCH MASK
(729)	1833	BITSTRING	2		UNUSED
(72B)	1835	BITSTRING	1	DYOMDUP	SAVE DUPLICATE COUNT
(72C)	1836	ADDRESS	4	DYOMNUM	SAVE MASTER NUMBER
SPOOL MANAGER WORK AREA (X-PARTITION I/F)					
(6B0)	1712	CHARACTER	8	DYJN	SPOOL MANAGEMENT JOB NAME
(6B8)	1720	BITSTRING	2	DYJ#	SPOOL MANAGEMENT JOB NO
(6BA)	1722	BITSTRING	1	DYFG	FLAG BYTE COPIED FROM PIB
		1...		DYVM	"X'80'" .. VIRTUAL MODE
(6BB)	1723	BITSTRING	1	DYSW	SWITCH BYTE
(6BC)	1724	SIGNED	4	DYXA	TASK ERR EXIT RTN ADDR
XPCC CROSS PARTITION USER TASK RE-DEFINITION					
(6B0)	1712	ADDRESS	4	DYXTIML	TIME LIMIT
(6B4)	1716	ADDRESS	4	DYXXPCC	ADDRESS OF XPCCB BEING USED
(6B8)	1720	ADDRESS	4	DYXWRKA	ADDRESS OF WORKAREA
(6BC)	1724	ADDRESS	4	DYXEDCB	ADDRESS OF ASSOCIATED EDCB
(6D0)	1744	ADDRESS	4	DYXCKPA	ADDR OF EXT CKP INFO
(6D4)	1748	SIGNED	2	DYXCKPL	LENGTH OF EXT CKP INFO
(6D6)	1750	SIGNED	2		UNUSED
(6D8)	1752	ADDRESS	4	DYACIET	$$XTM: ADDR. OF TMP. ACIE
(6DC)	1756	ADDRESS	4	DYACITQ	$$XTM: ADDR. OF TQE
(66C)	1644	ADDRESS	4	DYXSPL	ADDRESS OF ASSOCIATED SPL
		1111 111.		DYXSID	"X'FE'" .. XPCC SPL PRESENT
LOGICAL OUTPUT SPOOLER RE-DEFINITION					
(690)	1680	ADDRESS	4	DYOSNR	SAVED RECORD COUNT
(694)	1684	ADDRESS	4	DYOSLC	SAVED LINE/CARD COUNT
(698)	1688	SIGNED	4	DYOSPC	SAVED PAGE COUNT
(69C)	1692	SIGNED	2	DYOSNT	SAVED NO OF TRACKS/BLOCKS
PRINT STATUS TASK RE-DEFINITION (QUEUE DISPLAY)					
(6B0)	1712	ADDRESS	4	DYPSQN	NEXT QUEUE SET NUMBER
(6B0)	1712	ADDRESS	4	DYPSJCA	POFFLOAD JOURNALING JCA PNTR

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(6B4)	1716	ADDRESS	4	DYPSWA	ADDRESS OF PS WORKAREA
(6B8)	1720	BITSTRING	1	DYPSFG	FLAG BYTE
(6B9)	1721	CHARACTER	7	DYPSLB	BAM LABEL IF ANY
(6C0)	1728	SIGNED	2	DYPSOFTP	PDISPLAY TAPE OFTAP=
(6C2)	1730	BITSTRING	2		UNUSED
(6C4)	1732	SIGNED	4		UNUSED 21
(6C8)	1736	BITSTRING	4		UNUSED
	PSTART RDR/LST/PUN TAPE TASK DEFINITION				
(6B0)	1712	CHARACTER	7	DYTKLB	BAM LABEL IF ANY
(6B7)	1719	BITSTRING	1		UNUSED
	RJE,BSC TASK RE-DEFINITION				
(6C0)	1728		0	DYBQ	"DYRS" BSC APPENDAGE CHAIN PTR
(6C0)	1728		0	DYLRQ	"DYCT+8" LCB-TO-RELEASE-QUEUE
(6B0)	1712	ADDRESS	4	DYBS1S	LINKAGE-REG SAVE AREA
(6B4)	1716	ADDRESS	4	DYBS2S	2ND LINKAGE-REG SAVE AREA
(6B8)	1720	ADDRESS	4	DYBS3S	3.RD LEVEL LINKAGE SAVE
(5D4)	1492	ADDRESS	4	DYSR	SYSREC HEADER
	INITIALIZATION TASK RE-DEFINITION				
(6E0)	1760	ADDRESS	4	DYI4RTN	RETURN ADDRESS USED BY $$AT IF $$I4 OPEN IJDTEST FAILS
(6E0)	1760	ADDRESS	4	DYI3RTN	RETURN ADDRESS USED BY $$AT IF $$I3 OPEN IJQFILE AND IJQTEST FAILS. SAME ADDR. ALLOWS REUSE OF $$AT CODE.
(768)	1896		0	DYEN	"*" END OF STANDARD TCB
(768)	1896		0	DYLN	"*-DYSD" LENGTH OF TCB
	TCB-EXPANSION FOR SAVE-ACCOUNT TASK				
(768)	1896	CHARACTER	7	DYSAFN	TAPE/DASD FILE NAME
(76F)	1903	CHARACTER	1	DYSADY	TAPE DENSITY ... ...REFER ALSO TO TCF8MS
(770)	1904	CHARACTER	4	DYSADV	DEVICE WHERE TO SAVE
(774)	1908	ADDRESS	4	DYSAPB	PUB-ADDR DEV WHERE TO SAVE
(778)	1912	ADDRESS	4	DYSAR1	DEVICE DATA PASSED FROM CP
(77C)	1916	ADDRESS	4	DYSART	LINKAGE-REG SAVE AREA
(780)	1920	ADDRESS	4	DYSARN	2ND LINKAGE-REG SAVE AREA
(784)	1924	ADDRESS	4	DYSADP	DTF-POINTER
(784)	1924		0	DYLN1	"*-DYSD" LENGTH TCB INCL. SAVE-ACC.
		..1.		DYLN2	"*-DYSAFN" LENGTH OF EXPANSION
	OVERLAY FOR PNET TASKS				
(690)	1680	ADDRESS	4	DYENCB	ADDRESS OF NODE CTRL BLOCK
(694)	1684	ADDRESS	4	DYENTE	ADDR OF NCB TASK ENTRY
(698)	1688	BITSTRING	1	DYERCB	RCB OF TASK CONCERNED
(699)	1689	BITSTRING	1	DYETTC	TERMINATION CONDITION BYTE
		1...		DYETSO	"X'80'" .. SIGNOFF RECORD SENT/REC
		.1..		DYETLC	"X'40'" .. LINE ERROR STOP
(69A)	1690	BITSTRING	2	DYEFCS	FCS BYTES
(69C)	1692	ADDRESS	4	DYEWKA	ADDRESS OF WORKAREA
(6A0)	1696	BITSTRING	1	DYEST1	STATUS BYTE 1
		1...		DYERIF	"X'80'" .. RIF SENT/RECEIVED
		.1..		DYEPGR	"X'40'" .. PERMISSION GRANTED SENT/RECEIVED
		..1.		DYEPRJ	"X'20'" .. PERMISSION REJECTED SENT/RECEIVED
		..1.		DYERCS	"X'20'" .. RECEIVER CANCEL SENT/RECEIVED
		...1		DYEEOF	"X'10'" .. EOF SENT/RECEIVED
		.... 1...		DYEADS	"X'08'" .. ABORT TRANSMISSION SENT/RECEIVED
		.... .1..		DYECMC	"X'04'" .. TRANSMISSION COMPLETE SENT/RECEIV
(6A1)	1697	BITSTRING	1	DYEST2	STATUS BYTE 2
		1...		DYEWIB	"X'80'" .. WAITING FOR INPUT BUFFER

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
		.1..		DYECRTL	"X'40'" .. COMPRESSION ERROR
		..1.		DYENOP	"X'20'" .. DON'T POST THIS TASK
		...1		DYESPD	"X'10'" .. TASK SUSPENDED
		.... 1...		DYEPBO	"X'08'" .. POST ONLY AFTER BUFFER SENT
		.... .1..		DYERCA	"X'04'" .. RECEIVER CANCEL AFTER ABORT SENT
		.... ..1.		DYERAB	"X'02'" .. RELEASE OF ALL BUFFERS REQUESTED
		1		DYESOB	"X'01'" .. SHORT ON BUFFER CONDITION
(6A2)	1698	BITSTRING	2		UNUSED
PART FOR RECEIVER TASK					
(6A4)	1700	ADDRESS	4	DYERHD	ADDR OF RECEIVED INPUT BUFFER QUEUE
(6A8)	1704	ADDRESS	4	DYERTL	TAIL PTR RECEIVED INPUT BUFFER QUEUE
(6AC)	1708	BITSTRING	1	DYENRB	NUMBER OF RECEIVED BUFFERS
(6AD)	1709	BITSTRING	3		UNUSED
(6B0)	1712	BITSTRING	4		UNUSED
(6B0)	1712		0	DYELN	"*-DYSD" LENGTH EXTENDED TCB FOR PNET
OVERLAY FOR TRANSMITTER TASK					
(6A4)	1700	ADDRESS	4	DYEFOB	ADDR OF FREE OUTPUT BUFFER QUEUE
(6A8)	1704	BITSTRING	1	DYENAB	NUMBER OF ACQUIRED BUFFERS
(6A9)	1705	BITSTRING	3		UNUSED
(6AC)	1708	SIGNED	4	DYETL#	TOTAL LINE NUMBER
(6B0)	1712	SIGNED	4	DYECL#	CURRENT LINE NUMBER
MISCELLANEOUS					
(788)	1928	SIGNED	2	OURPIK	VSE/POWER PIK
(78A)	1930	SIGNED	2	OTHPIK	PIK OF PARTITION CONTROLLED BY VSE/P
(78C)	1932	ADDRESS	4	OTHPCE	POINTER TO PCE OF PARTITION
VARIABLES USED FOR SUBROUTINE 'BINTODEC'					
(790)	1936	DBL WORD	8	DBLWRD	DOUBLE WORD USED FOR CONVERSION
(798)	1944	SIGNED	4	CONVBIN	CONTAINS BINARY # TO BE CONVERTED
(79C)	1948	CHARACTER	10	CONVDEC	CONTAINS DECIMAL NUMBER IN PRINTABLE FORMAT
(7A6)	1958	CHARACTER	2		RESERVED FOR FUTURE USE
VARIABLES USED FOR SUBROUTINE 'INVDEV' OR 'PASSIGN' OR 'VPARTID'					
(7A8)	1960	CHARACTER	4	CPWASDEV	KIND OF DEVICE WANTED
(7AC)	1964	BITSTRING	1	CPWASDTY	PUB DEVICE TYPE CODE
(7AD)	1965	BITSTRING	2	CPWASCUU	CUU OF DEV IN BIN(PACKED) FORMAT
(7AF)	1967	BITSTRING	1	CPWASFB1	ARGUMENT FLAG BYTE 1
		111. ..11		DEVTAPE	"C'T'" .. TAPE SPOOLING REQUESTED
(7B0)	1968	ADDRESS	4	CPWASPBP	ADDR OF DEVICE PUB ENTRY
(7B4)	1972	BITSTRING	1	CPWASLUN	LOGICAL UNIT NUMBER INDEX
(7B5)	1973	BITSTRING	1	CPWAMODE	TAPE MODE FOR ASSIGN RTN.
(7B6)	1974	BITSTRING	2	CPWASAVE	SAVE FIELD FOR ALTERN. DEVICE
(7B8)	1976	BITSTRING	2	CPWVPSLG	SYSLOG ID OF PARTITION
(7BA)	1978	BITSTRING	2		RESERVED FOR FUTURE USE
VARIABLES USED FOR SUBROUTINE 'TCBSCAN'					
(7BC)	1980	ADDRESS	4	CPWTCBPT	CURRENT TCB POINTER
THE FOLLOWING ARGUMENTS ARE SET UP BY THE CALLING ROUTINE IN ORDER TO LOCATE THE TCB MEETING ALL APPLICABLE CRITERIA. WHEN AN ARGUMENT CONTAINS HEX ZEROS, IT IS ASSUMED TO BE NOT IMPORTANT AND WILL BE NOT CHECKED.					
(7C0)	1984	CHARACTER	16	CPWTARG (0)	ARGUMENT LIST
(7C0)	1984	CHARACTER	1		RESERVED
(7C1)	1985	CHARACTER	3	CPWTACU	DEVICE ADDRESS 'CUU'

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(7C4)	1988	CHARACTER	4	CPWTATI	TASK IDENTIFIER
(7C8)	1992	BITSTRING	1	CPWTABIN	BINARY RJE USER-ID
(7C9)	1993	CHARACTER	3	CPWTADEC	PRINTABLE DECIMAL RJE USER-ID
(7CC)	1996	BITSTRING	1	CPWTAFLG	ARGUMENT SWITCH BYTE
		1...		CPWTACNT	"X'80'" .. INDICATES THAT THE SCAN SHOULD NOT BE STARTED AT THE BEGINNING OF THE TCB-CHAIN, BUT AT THE POINT WHERE THE PREVIOUS SCAN STOPPED.
(7CD)	1997	BITSTRING	3		RESERVED FOR FUTURE USE

VARIABLES USED FOR SUBROUTINE 'CLASS'
THE FOLLOWING ARGUMENTS ARE SET UP BY THE CALLING ROUTINE
IN ORDER TO BUILD THE CLASS POINTER'S IN THE DUMMY TCB AREA

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(7D0)	2000	CHARACTER	12	CPWCLAR (0)	ARGUMENT LIST
(7D0)	2000	CHARACTER	6	CPWCLAS	CLASS(ES) UP TO 4 CLASSES
(7D6)	2006	BITSTRING	1	CPWCLIX	USED FOR TRANSLATION OF CLASS CHAR.
(7D7)	2007	BITSTRING	1	CPWCL#C	MAX. NUMBER OF VALID CLASSES - 4 FOR EXECUTION READER TASKS - 4 FOR PHYSICAL WRITER TASKS - 1 FOR PHYSICAL READER TASKS
(7D8)	2008	CHARACTER	3	CPWCLTI	TASK TYPE ('LST', 'RDR' OR 'PUN')
(7DB)	2011	BITSTRING	1	CPWCLDF	DEFAULT CLASS

VARIABLES USED FOR SUBROUTINE 'QRINSPCT'
THE FOLLOWING ARGUMENTS ARE SET UP BY THE CALLING ROUTINE
IN ORDER TO DETERMINE IF A QUEUE SET MEETS ALL APPLICABLE
CRITERIA.
WHEN AN ARGUMENT CONTAINS HEX ZEROS, IT IS ASSUMED TO BE NOT
IMPORTANT AND WILL BE NOT CHECKED.

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(7DC)	2012	CHARACTER	170	CPWQARG	ARGUMENT LIST
(7DC)	2012	CHARACTER	8	CPWQAJN	JOB NAME
(7E4)	2020	ADDRESS	2	CPWQAJ#	JOB NUMBER
(7E6)	2022	BITSTRING	1	CPWQACL	CLASS ASSOCIATED TO QUEUE SET
(7E7)	2023	BITSTRING	1	CPWQAF1	FLAG BYTE 1
		1...		CPWQAF1S	"X'80'" ..JOB SUFFIX MUST BE THERE
		.1..		CPWQAF1C	"X'40'" ..C-TYPE PARAMETER SPECIF'D
		..1.		CPWQAF1D	"X'20'" ..CDUE=* SPECIFIED
(7E8)	2024	BITSTRING	1	CPWQABIN	BINARY RJE USER-ID (0 FOR CENTRAL OP
(7E9)	2025	CHARACTER	3	CPWQADEC	PRINTABLE DECIMAL RJE USER-ID
(7EC)	2028	CHARACTER	8	CPWQAGJN	GENERIC JOB NAME
(7F4)	2036	BITSTRING	1	CPWQAGLN	LENGTH OF GENERIC SUPPLIED JOBNAME
(7F5)	2037	BITSTRING	1	CPWQAPCB	QUEUE PROCESSING FLAGS
(7F6)	2038	ADDRESS	4	CPWQQNUM	QUEUE ENTRY NUMBER
(7FA)	2042	BITSTRING	2		RESERVED FOR FUTURE USE
(7FC)	2044	CHARACTER	3	CPWQACT1 (0)	C-TYPE OPERANDS 1
(7FC)	2044	CHARACTER	1	CPWQACDP	CURRENT DISPOSITION
(7FD)	2045	CHARACTER	1	CPWQACPY	CURRENT PRIORITY
(7FE)	2046	CHARACTER	1	CPWQACSY	CURRENT SYSID
(7FF)	2047	BITSTRING	1	CPWQAJSF	JOB SUFFIX NUMBER
(800)	2048	CHARACTER	36	CPWQACT2 (0)	C-TYPE OPERANDS 2
(800)	2048	CHARACTER	8	CPWQACNN	CURRENT 'TO' NODE NAME

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(808)	2056	CHARACTER	8	CPWQACUS	CURRENT 'TO' USER ID
(810)	2064	CHARACTER	4	CPWQACFI	CURRENT FORMS ID (FNO)
(814)	2068	CHARACTER	8	CPWQAFNN	'FROM' NODE NAME
(81C)	2076	CHARACTER	8	CPWQAFUS	'FROM' USER ID
(824)	2084	CHARACTER	8	CPWQAWRK	WORK FIELD
(82C)	2092	CHARACTER	8	CPWQADTE	CURRENT DATE 'CCYYMMDD'
(834)	2100	BITSTRING	1	CPWQAMSK	BRANCH MASK
(835)	2101	CHARACTER	8	CPWQUSER	'FROM' OR 'TO' USER ID
(83D)	2109	BITSTRING	1	CPWQCIX	PDISPLAY CLASS INDEX
(83E)	2110	CHARACTER	1	CPWQXSBQ	XMT SUBQUEUE, USED BY $$CA, $$CL, $$CR, $$CH CMD'S
(83F)	2111	BITSTRING	1		RESERVED FOR FUTURE USE
(840)	2112	BITSTRING	16	CPWQACUI	CURRENT USER INFO
(850)	2128	BITSTRING	4	CPWQ#QE	NUMBER OF QUEUE ENTRIES ADDRESSED BY COMMAND
(854)	2132	BITSTRING	8		RESERVED
(85C)	2140	BITSTRING	4	CPWQACPG	CPAGES LIMIT VALUE
(860)	2144	BITSTRING	4	CPWQACCD	CCARDS LIMIT VALUE
(864)	2148	BITSTRING	1	CPWQAPMS	CPAGES BRANCH MASK
(865)	2149	BITSTRING	1	CPWQACMS	CCARDS BRANCH MASK
(866)	2150	BITSTRING	32		RESERVED

FOR ANY FURTHER EXTENSION, UPDATE LENGTH OF 'CPWQARG'
AND RE-COMPILE IPW$$CM TO CLEAR PERM. CMD. WORK-AREA
RE-COMPILE ALL CP MODULES ACCESSING CP WORK AREA
VARIABLES USED FOR SUBROUTINE 'VQEUEID'

(888)	2184	ADDRESS	4	QCLASPTR	BEGIN OF CLASS TABLE
(88C)	2188	ADDRESS	4	CLASSPTR	POINTS TO ACTUAL CLASS
(890)	2192	SIGNED	2	CLASSLC	NUMBER OF SCANS TO BE PERFORMED
(892)	2194	CHARACTER	1	CLASSQID	QUEUE RECORD IDENTIFIER
(893)	2195	CHARACTER	1	CLASSPCB	QUEUE PROCESSING FLAGS

VARIABLES USED BY OPERAND FORMATTING ROUTINE

(894)	2196	ADDRESS	4	CMNDPTR	POINTS TO COMMAND TABLE ENTRY
(898)	2200	ADDRESS	4	DELIMPTR	ADDRESS OF CURRENT DELIMITER
(89C)	2204	BITSTRING	256	TRTTAB	TRANSLATE AND TEST TABLE
(99C)	2460	SIGNED	2	MAXOP	MAX NUMBER OF OPERANDS ALLOWED
(99E)	2462	SIGNED	2	OPNUM	CURRENT # OF OPERAND IN PROCESS
(9A0)	2464	CHARACTER	8	SVEOP	OPERAND SAVE FIELD

VARIABLES USED FOR QUEUE MANIPULATION

(9A8)	2472	ADDRESS	4	CPWQMSA	SAVE FIELD
(9AC)	2476	ADDRESS	4		RESERVED
(9B0)	2480	BITSTRING	1	CPWQFLG	FLAG BYTE
		1...		CPWQFND	"X'80'" PROCESSING NON-DISP CHAIN
(9B1)	2481	BITSTRING	3		UNUSED

VARIABLES USED BY PSTART COMMAND PROCESSOR
THE FOLLOWING FIELDS ARE USED TO BUILD THE PARTITION CONTROL BLOCK.

(9B4)	2484		0	CPWSTRT	"*"
(9B4)	2484	CHARACTER	1	CPWPDOC	DEFAULT OUTPUT CLASS
(9B5)	2485	BITSTRING	1	CPWPDMT	MULTI TASK INDICATOR
(9B6)	2486	BITSTRING	2		RESERVED, UNUSED

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(9B8)	2488	ADDRESS	4	CPWSPCE	CURRENT TEMPORARY SPOOL ENTRY
(9BC)	2492	SIGNED	2	CPWSP#E	NUMBER OF ENTRIES IN SPOOL TABLE
(9BE)	2494	SIGNED	2	CPWSP#O	OLD NUMBER OF ENTRIES
(9C0)	2496	ADDRESS	4	CPWSPCO	OLD SPOOL TABLE POINTER
(9C4)	2500	BITSTRING	1	CPWSPF1	FLAG BYTE 1
		1...		CPWSPGE	"X'80'" .. READER ENTRY GENERATED (WRITER ONLY PARTITION)
		.1..		CPWSNPC	"X'40'" .. "NPC" OPTION SPECIFIED
(9C5)	2501	BITSTRING	1		RESERVED FOR FUTURE USE
(9C6)	2502	BITSTRING	58	CPWSPDE	TEMPORARY SPOOL TABLE (CONTAINING TWO BYTE PUB TABLE INDECES)
(A00)	2560	BITSTRING	1		RESERVED FOR FUTURE USE
(A04)	2564	ADDRESS	4	CPWSPRE	SAVE AREA FOR CALLER'S RETURN ADDR
(A08)	2568	ADDRESS	4	CPWSTPN	ADDRESS TO PHASE NAME SUFFIX
(A0C)	2572	BITSTRING	4	CPWSTRF	MACRO ID AND RETURN CODE
MAP OF PARTITION BOUNDARY INFORMATION					
(A10)	2576	ADDRESS	4	CPWPAVB	VIRTUAL PARTITION BEGIN ADDRESS
(A14)	2580	ADDRESS	4	CPWPAGB	VIRTUAL GETVIS AREA START ADDRESS
(A18)	2584	ADDRESS	4	CPWPAVE	VIRTUAL PARTITION END ADDRESS
(A1C)	2588	ADDRESS	4	CPWPARB	REAL PARTITION START ADDRESS
(A20)	2592	ADDRESS	4	CPWPARE	REAL PARTITION END ADDRESS
		...1 .1..		CPWPALN	"*-CPWPAVB" LENGTH OF BOUNDARY INFORMATION
SAVE AREAS FOR PSTART/PNET OPERANDS					
(9B4)	2484		0	CPWSN	"*"
(9B4)	2484	CHARACTER	3	CPWSLADR	LINE ADDRESS AS CUU
(9B7)	2487	CHARACTER	8	CPWSLPAS	LINE PASSWORD
(9BF)	2495	CHARACTER	8	CPWSNDID	NODE ID
(9C7)	2503	CHARACTER	8	CPWSNDPW	NODE PASSWORD
(9CF)	2511	BITSTRING	1	CPWTRFLG	TRACE FLAG
		.... ..1.		CPWTRFL1	"X'02'" TRACE ACTIVE
(9D0)	2512	BITSTRING	1	CPWSNAFG	SNA NODE FLAG
(9D1)	2513	BITSTRING	1	CPWTYPF	UPDATES NCBTYP (BSC,CTC,RES)
		...1 111.		CPWSNL	"*-CPWSN" LENGTH OF WORKAREA
VARIABLES USED BY - PSTART/PSTOP TASKTR AND TRACE FACILITY - PVARY COMMAND					
(9B4)	2484	ADDRESS	4	CPWTTSIZ	TRACE AREA SIZE IN BYTES
(9B8)	2488	ADDRESS	4	CPWTFCS	COMPARE-AND-SWAP WORKAREA
(9BC)	2492	BITSTRING	1	CPWTTFLG	FLAG BYTE
		1...		CPWTTFEN	"X'80'" .. TASK TRACING ENABLED
		.1..		CPWTTTTR	"X'40'" .. TASK TRACE SPECIFIED
		..1.		CPWTTTEN	"X'20'" .. TRANSMITTER EXIT ENABLED
		...1		CPWTTOEN	"X'10'" .. OUTPUT EXIT SPECIFIED
		.... 1...		CPWTTREN	"X'08'" .. JOB EXIT SPECIFIED
		.... .1..		CPWTTPEN	"X'04'" .. PNET EXIT SPECIFIED
		.... ..1.		CPWTTXEN	"X'02'" .. EXIT/IGNREC ENABLE WANT'D
		1		CPWTTIGN	"X'01'" .. IGNORE RECORDING REQUEST
(9BD)	2493	BITSTRING	1	CPWTTFG2	FLAG BYTE 2
		1...		CPWTT2TF	"X'80'" ..TRACE FACILITY SPECIFIED
		.1..		CPWTT2FT	"X'40'" ..FULL TASK TRACING
(9BE)	2494	BITSTRING	1	CPWTVFLG	PVARY DYNC FLAG BYTE
		1...		CPWTVFVD	"X'80'" DYNC VARIATION DONE
		.1..		CPWTVFEN	"X'40'" ENABLE DYNAMIC CLASSES
		..1.		CPWTVFDI	"X'20'" DISABLE DYNAMIC CLASSES
		...1		CPWTVFAL	"X'10'" ADDRESS ALL CLASSES
		.... 1...		CPWTVFCL	"X'08'" ADDRESS SELECTED CLASSES
		.... .1..		CPWTVMGI	"X'04'" 'INVALID' MESSAGE ISSUED
		.... ..1.		CPWTVALI	"X'02'" ALL CLASSES FOUND INVALID
(9BF)	2495	BITSTRING	23	CPWTVCOL	PVARY ALL CLASS COLLECTION

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(9D6)	2518	BITSTRING	1	CPWTVCOE	CLASS COLLECTION END INDIC.
(9D7)	2519	CHARACTER	7	CPWTFTBL	PSTART TRACE TABLE=NAME
		VARIABLES USED BY PLOAD COMMAND			
(9B4)	2484	ADDRESS	4	CPWY012 (3)	SAVE AREA FOR REG. 0,1,2
(9C0)	2496	SIGNED	2	CPWXSIZE	WORK AREA SIZE FOR EXIT
(9C2)	2498	BITSTRING	1	CPWYOPT	PLOAD DYNCTAB OPTIONS
		1...		CPWYOCO	"X'80'" .. PLOAD DYNCTAB COND
		.1..		CPWYOFO	"X'40'" .. PLOAD DYNCTAB FORCE
		..1.		CPWYOVE	"X'20'" .. PLOAD DYNCTAB VERIFY
		...1		CPWYOLT	"X'10'" .. PLOAD DYNCTAB,,LST
		.... 1...		CPWYXPU	"X'08'" .. RUN EXIT AS 'PA' WORKUNIT
(9C3)	2499	CHARACTER	1	CPWYCLC	DYNAMIC CLASS TABLE INDEX
		VARIABLES USED BY PSTOP, PFLUSH, PFLUSH/PNET COMMAND PROCESSOR			
(9B4)	2484	CHARACTER	1	CPWTTC	STOP CODE
(9B5)	2485	CHARACTER	3		RESERVED FOR FUTURE USE
(9B8)	2488	CHARACTER	8	CPWLUNM	LOGICAL UNIT NAME
(9C0)	2496	CHARACTER	3	CPWRVTRL	TRANSMITTER/RECEIVER LINE #
(9C3)	2499	CHARACTER	1		NOT USED
(9C4)	2500	CHARACTER	8	CPWFNOD	NODE ID
		VARIABLES USED BY PSTART/PSTOP ... DEV COMMAND			
(9B4)	2484	CHARACTER	8	CPWXDEV	DEVICE NAME
(9BC)	2492	CHARACTER	8	CPWXDDS	DDS NAME
(9C4)	2500	CHARACTER	1	CPWXTTC	TERMINATION CODE
(9C5)	2501	BITSTRING	1	CPWXFLG	LENGTH OF PARM FIELD
(9C6)	2502	BITSTRING	2		RESERVED
(9C8)	2504	CHARACTER	60	CPWXPARM	PARAMETER FIELD
		IPW$DCP CPW=NO,PSWKSP=YES GENERATE PDISPLAY ARG LIST PDISPLAY ARGUMENT LIST THE FOLLOWING FIELDS REPRESENT THE ARGUMENT LIST WHICH WILL BE PASSED TO THE PRINT STATUS TASK TO PERFORM THE APPROPRIATE DISPLAY FUNCTIONS. THE FIRST 28 BYTES ARE USED AS COMMOM PART BY PDISPLAY XX AS WELL AS BY PDISPLAY PNET OR DYNC.			
(9B4)	2484	SIGNED	4	CPWDARGL (0)	ARGUMENT LIST
(9B4)	2484	CHARACTER	1	CPWDID	PARAMETER LIST FLAG BYTE
		11.. .1..		CPWDDID	"C'D'" .. ID FOR DEFAULT DISPLAY
		11.1 .111		CPWDPID	"C'P'" .. ID FOR PNET DISPLAY
		11.1 1...		CPWDVID	"C'Q'" .. ID FOR CORE COPY DISPLAY
		111. ..11		CPWDTID	"C'T'" .. ID FOR TAPE DISPLAY
		111. 1...		CPWDYID	"C'Y'" .. ID FOR DYNC DISPLAY
		111. ..1.		CPWDSID	"C'S'" .. ID FOR STATISTICS DISPLAY
		11.. .1.1		CPWDEID	"C'E'" .. ID FOR EXIT DISPLAY
		11.1 ...1		CPWDJID	"C'J'" .. ID FOR POFFLOAD JOURNAL @*
(9B5)	2485	BITSTRING	1	CPWDTOID	DESTINATION OF REPORT
(9B6)	2486	BITSTRING	2	CPWDLU	LOGICAL UNIT # OF PRINTER
(9B8)	2488	ADDRESS	4	CPWDECB	ECB ADDRESS
(9BC)	2492	ADDRESS	4	CPWDTCB	ADDR OF REQUESTING TASK TCB

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(9C0)	2496	CHAR-ACTER	1	CPWDCCL	DYN.CLASS FOR PDISPLAY CRE
(9C0)	2496	CHAR-ACTER	2	CPWDCPI	PART.ID FOR PDISPLAY CRE
(9C2)	2498	BITSTRING	1	CPWDFLG1	FLAG1 BITS:
		1...		CPWD1DUE	"X'80'" - PDISPLAY CDUE=*
		.1..		CPWD1FRR	"X'40'" - PDISPLAY RDR,FREER
		..1.		CPWDDEX	"X'20'" - PDISPLAY EXIT DATA FND
		...1		CPWD1PAR	"X'10'" - PDISPLAY CRE,PART
		.... 1...		CPWD1CPI	"X'08'" - PDISPLAY CRE,PART,PID
		.... .1..		CPWD1CCL	"X'04'" - PDISPLAY CRE,PART,CLS
		.... ..1.		CPWD1ALL	"X'02'" - PDISPLAY CRE,ALLSYS
(9C3)	2499	BITSTRING	1	CPWDFLG	FLAG BITS:
		1...		CPWDREM	"X'80'" - PDISPLAY RJE
		.1..		CPWDHLD	"X'40'" - PDISPLAY HOLD
		..1.		CPWDFRE	"X'20'" - PDISPLAY FREE
		...1		CPWDLOC	"X'10'" - PDISPLAY LOCAL
		.... 1...		CPWDCON	"X'08'" - DISPLAY TARGET = CON
		.... .1..		CPWDPRT	"X'04'" - DISPLAY TARGET = PRT
		.... ..1.		CPWDLST	"X'02'" - DISPLAY TARGET = SPOOL LST QUEUE ENTRY
		1		CPWDFUL	"X'01'" - PDISPLAY ..FULL=YES
(9C4)	2500	CHAR-ACTER	9	CPWDFNM (0)	FROM NODE NAME + SYSID
(9C4)	2500	CHAR-ACTER	8	CPWDFNMN	.. FROM NODE NAME
(9CC)	2508	CHAR-ACTER	1	CPWDFNMS	.. FROM SYSID
(9CD)	2509	CHAR-ACTER	8	CPWDUID	FROM USER/REMOTE ID
(9D5)	2517	BITSTRING	1	CPWDNMRF	FLAG BYTE FROM NMR
(9D6)	2518	BITSTRING	1	CPWDFG2	COPY OF CPFG2 FLAG BYTE
(9D7)	2519	BITSTRING	1		RESERVED
(9D8)	2520	SIGNED	4	CPWDPPA	COPY OF $ICP PASS VALUE
(9DC)	2524	ADDRESS	4	CPWDBS	BEGIN SCAN INDICATOR
(9E0)	2528	BITSTRING	1	CPWDQID	QUEUE PROCESSING FLAGS
		1...		CPWDQIDR	"X'80'" .. RDR QUEUE DISPLAY @*
		.1..		CPWDQIDL	"X'40'" .. LST QUEUE DISPLAY @*
		..1.		CPWDQIDP	"X'20'" .. PUN QUEUE DISPLAY @*
		...1		CPWDQIDX	"X'10'" .. XMT QUEUE DISPLAY @*
		.... 1...		CPWDQIDW	"X'08'" .. WFR SUBQUEUE DISPLAY
		.... .1..		CPWDQIDD	"X'04'" .. DELAYED DELETE DISPLAY
		.... ..1.		CPWDQIDC	"X'02'" .. CREATE QUEUE DISPLAY
		1		CPWDQIDB	"X'01'" .. BIGGEST QUEUE DISPLAY
(9E1)	2529	BITSTRING	2	CPWDJN	JOBNUMBER
(9E3)	2531	BITSTRING	1	CPWDBIN	REMOTE ID (BINARY FORMAT)
(9E4)	2532	BITSTRING	1	CPWDGJL	LENGTH OF GENERIC JOBNAME
(9E5)	2533	CHAR-ACTER	8	CPWDGJN	GENERIC JOBNAME
(9ED)	2541	CHAR-ACTER	8	CPWDJOB	JOBNAME
(9F5)	2549	BITSTRING	8	CPWDTNN	TARGET NODE NAME
(9FD)	2557	BITSTRING	1	CPWDCDP	CURRENT DISPOSITION
(9FE)	2558	BITSTRING	1	CPWDCPY	CURRENT PRIORITY
(9FF)	2559	BITSTRING	1	CPWDCSY	CURRENT SYSTEM ID
(A00)	2560	CHAR-ACTER	4	CPWDCFI	CURRENT FORMS ID (FNO)
(A04)	2564	CHAR-ACTER	8	CPWDTUS	CURRENT 'TO' USER ID
(A0C)	2572	CHAR-ACTER	8	CPWDFNN	'FROM' NODE NAME
(A14)	2580	CHAR-ACTER	8	CPWDFUS	'FROM' USER ID
(A1C)	2588	CHAR-ACTER	1	CPWDCLS	JOBCLASS
(A1D)	2589	BITSTRING	1	CPWDCIX	PDISPLAY CLASS INDEX

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(A1E)	2590	BITSTRING	2		RESERVED
(A20)	2592	CHARACTER	8	CPWDWRK	WORK FIELD
(A28)	2600	CHARACTER	8	CPWDDTE	CURRENT DATE 'CCYYMMDD'
(A30)	2608	BITSTRING	1	CPWDMSK	BRANCH MASK
(A31)	2609	CHARACTER	8	CPWDUSER	'FROM' OR 'TO' USER ID
(A39)	2617	BITSTRING	1	CPWDCQID	CURRENT PROCESSED QUEUE $$PS
(A3A)	2618	BITSTRING	2		RESERVED
(A3C)	2620	ADDRESS	4	CPWDTPUB	TAPE UNIT PUB ENTRY ADDR
(A40)	2624	SIGNED	2	CPWDTPUU	TAPE UNIT PROG.LOG.NUMBER
(A42)	2626	CHARACTER	3	CPWDTCUU	TAPE UNIT CUU (EBCDIC)
(A45)	2629	BITSTRING	3		RESERVED
(A48)	2632	ADDRESS	4	CPWDPPUB	PRT UNIT PUB ENTRY ADDR
(A4C)	2636	SIGNED	2	CPWDPPUU	PRT UNIT PROG.LOG.NUMBER
(A4E)	2638	CHARACTER	3	CPWDPCUU	PRT UNIT CUU (EBCDIC)
(A51)	2641	BITSTRING	3		RESERVED
(A54)	2644	ADDRESS	4	CPWSDSPY	SAVE ADDRESS DST BLOCKS
(A58)	2648	CHARACTER	16	CPWDCUIN	CURRENT USER INFO
(A68)	2664	BITSTRING	4	CPWDCPG	CPAGES LIMIT VALUE
(A6C)	2668	BITSTRING	4	CPWDCCD	CCARDS LIMIT VALUE
(A70)	2672	BITSTRING	1	CPWDPMS	CPAGES BRANCH MASK
(A71)	2673	BITSTRING	1	CPWDCMS	CCARDS BRANCH MASK
(A72)	2674	BITSTRING	2	CPWDLIMT	PDISPLAY LIMIT VALUE
(A74)	2676	BITSTRING	2	CPWDLIMA	ACCUMULATED LIMIT VALUE
(A76)	2678	BITSTRING	28		RESERVED
		11.1 111.		CPWDARLN	"*-CPWDARGL" ARGUMENT LIST LENGTH

THE FOLLOWING ARE WORK FIELDS FOR PDISPLAY PARAMETERS
WHICH REQUIRE CONTEXT CHECKING BEFORE FINAL
PROCESSING

DEVICES STAGE 1 - SYNTAX CHECKING

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(A94)	2708	SIGNED	4	(0)	ALLIGNMENT
(A94)	2708	SIGNED	2	CPS1TCUX	POSSIBLE CUU(PACKED) OF TAPE
(A96)	2710	CHARACTER	3	CPS1TCUU	POSSIBLE CUU(EBCDIC) OF TAPE
(A99)	2713	BITSTRING	1		RESERVED
(A9A)	2714	SIGNED	2	CPS1PCUX	POSSIBLE CUU(PACKED) OF PRT
(A9C)	2716	CHARACTER	3	CPS1PCUU	POSSIBLE CUU(EBCDIC) OF PRT
(A9F)	2719	BITSTRING	1	CPS1FLG	POSSIBLE FLAGS
		1...		CPS1FGNW	"X'80'" .. TAPE REWIND=NO SPEC'D
		.1..		CPS1FGRW	"X'40'" .. TAPE REWIND=YES SPED'D

DEVICES STAGE 2 - RESOURCE ASSIGNMENTS

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(AA0)	2720	ADDRESS	4	CPS2TPUB	POSSIBLE PUB ADDR OF TAPE
(AA4)	2724	CHARACTER	3	CPS2TCUU	POSSIBLE CUU(EBCDIC) OF TAPE
(AA7)	2727	BITSTRING	1		RESERVED
(AA8)	2728	SIGNED	2	CPS2TPUU	POSSIBLE PUU OF TAPE UNIT
(AAA)	2730	SIGNED	2		RESERVED
(AAC)	2732	CHARACTER	3	CPS2PCUU	POSSIBLE CUU(EBCDIC) OF PRT
(AAF)	2735	BITSTRING	1		RESERVED
(AB0)	2736	SIGNED	2	CPS2PPUU	POSSIBLE PUU OF PRINTER
(AB2)	2738	SIGNED	2		RESERVED

DISPLAY TARGET SPECIFICATION

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(AB4)	2740	CHARACTER	1	CPS1TGSP	PDISPLAY TARGET - SPECIFIED

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
		1...		CPS1TSC	"X'80'" .. CONSOLE
		.1..		CPS1TSP	"X'40'" .. PRINTER
		..1.		CPS1TSL	"X'20'" .. LST SPOOL ENTRY
(AB5)	2741	CHARACTER	1	CPS1TGDF	PDISPLAY TARGET - DEFAULT
(AB5)	2741		0	CPWDTRLN	"*-CPWDARGL" PDISPLAY TOTAL WORK AREA LENGTH ==NOTE== MUST BE < 513

VARIABLES USED BY PDISPLAY PNET COMMAND
THE FOLLOWING FIELDS TOGETHER WITH THE COMMON PART OF THE
ARGUMENT LIST IS PASSED TO THE PRINT STATUS TASK TO PERFORM
THE PDISPLAY PNET FUNCTIONS.

(9DC)	2524	SIGNED	4	CPWDPPTR	POINTER TO SPECIFIED NODEID
(9E0)	2528	BITSTRING	1	CPWPFLG1	FLAG BYTE1
		1...		CPWDPOWN	"X'80'" .. OWN NODE DISPLAY REQUEST
		.1..		CPWDPLIN	"X'40'" .. LINK DISPLAY REQUEST
		..1.		CPWDPNID	"X'20'" .. SPECIFIC NODE DISPL.REQ
		...1		CPWDPALL	"X'10'" .. ALL NODES DISPLAY REQUEST
(9E1)	2529	BITSTRING	1		RESERVED
(9E2)	2530	CHARACTER	8	CPWNODID	NAME OF NODE ID

VARIABLES USED BY PDISPLAY DYNC/STATUS COMMAND
AT
THE FOLLOWING FIELDS TOGETHER WITH THE COMMON PART OF THE
ARGUMENT LIST IS PASSED TO THE PRINT STATUS TASK TO PERFORM
THE 'PDISPLAY DYNAMIC CLASS TABLE' OR
THE 'PDISPLAY STATISTICS STATUS' FUNCTIONS

(9DC)	2524	SIGNED	4	CPWDDPPA	POINTER TO DCLT AREA
(9E0)	2528	SIGNED	4	CPWDDNUM	JOB NO. OF LIST-Q ENTRY
(9E4)	2532	SIGNED	4	CPWDDLMG	NO. OF TERMINATION MESSAGE
(9E8)	2536	BITSTRING	1	CPWDDFL1	FLAG BYTE WITH CPFG SETTING
(9E9)	2537	BITSTRING	1	CPWDDFL2	FLAG BYTE 2
		1...		CPWD2ALL	"X'80'" DISPLAY ALL
		.1..		CPWD2ENA	"X'40'" DISPLAY ALL ENABLED
		..1.		CPWD2DIS	"X'20'" DISPLAY ALL DISABLED
		...1		CPWD2INV	"X'10'" DISPLAY ALL INVALID
		.... 1...		CPWD2CLS	"X'08'" DISPLAY A SINGLE CLASS
		.... .1..		CPWD2ACT	"X'04'" DISPLAY ACTIVE DCLT
		.... ..1.		CPWD2ONE	"X'02'" ONE LINE DISPLAYED
		1		CPWD2HED	"X'01'" HEAD LINE ALREADY DISPLAYED
(9EA)	2538	BITSTRING	1	CPWDDFL3	FLAG BYTE 3
		1...		CPWD3SUS	"X'80'" DISPLAY SUSPENDED
(9EB)	2539	BITSTRING	1	CPWDDCLS	CLASS TO BE DISPLAYED
		...1		CPWDDLEN	"*-CPWDDPPA" LENGTH OF 'DYNC' VARIABLES

VARIABLES USED BY PDISPLAY A COMMAND

(9B4)	2484	CHARACTER	12	CPWDAARG (0)	DISPLAY ACTIVE ARGUMENT LIST
(9B4)	2484	BITSTRING	1	CPWDAFLG	FLAG BYTE
		1...		CPWDAFPN	"X'80'" .. DISPLAY PNET TASKS
		.1..		CPWDAFPA	"X'40'" .. DISPLAY EXECUTION TASKS
		..1.		CPWDAFLO	"X'20'" .. DISPLAY LOCAL TASKS
		...1		CPWDAFRJ	"X'10'" .. DISPLAY RJE TASKS
		.... 1...		CPWDAFXT	"X'08'" .. DISPLAY X-PARTITION TASKS
		.... .1..		CPWDAFEX	"X'04'" .. DISPLAY DEV SERVICE TASKS
		.... ..1.		CPWDAFIN	"X'02'" .. DISPLAY INTERNAL TASKS
(9B5)	2485	BITSTRING	1		RESERVED
(9B6)	2486	CHARACTER	2	CPWDAPID	PARTITION ID
(9B8)	2488	CHARACTER	8	CPWDANID	NODE NAME
(9C0)	2496	BITSTRING	1	CPWDATYP	FLAG BYTE FOR EXEC. TASKS
		11.. .1..		CPWDATYD	"C'D'" .. DISPLAY DYAMIC TASKS
		111. ..1.		CPWDATYS	"C'S'" .. DISPLAY STATIC TASKS

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(9C1)	2497	CHAR-ACTER	1	CPWDAPIC	DYNAMIC CLASS SELECTED
		VARIABLES USED BY PALTER COMMAND PROCESSOR VARIABLE CPWALTSG USED BY PSEGMENT COMMAND PROCESSOR THE FOLLOWING FIELDS CONTAIN THE NEW KEYWORD VALUES SPECIFIED IN THE PALTER COMMAND. WHEN A FIELD CONTAINS HEX ZEROS, IT IS ASSUMED THAT NO CHANGE OF THE APPROPRIATE FIELD IN THE QUEUE RECORD SHOULD BE DONE			
(9B4)	2484	CHAR-ACTER	64	CPWALTER (0)	ARGUMENT LIST
(9B4)	2484	CHAR-ACTER	8	CPWALTTN	NEW TARGET NODE NAME
(9BC)	2492	CHAR-ACTER	8	CPWALTTU	NEW TARGET USER ID
(9C4)	2500	BITSTRING	1	CPWALTPY	NEW PRIORITY, IF NOT HEX ZERO
(9C5)	2501	CHAR-ACTER	1	CPWALTDP	NEW DISPOSITION, IF NOT HEX ZERO
(9C6)	2502	CHAR-ACTER	1	CPWALTCL	NEW CLASS, IF NOT HEX ZERO
(9C7)	2503	BITSTRING	1	CPWALTNC	NEW COPY NUMBER, IF NOT HEZ ZERO
(9C8)	2504	CHAR-ACTER	4	CPWALTCP	NEW COMPACTION TABLE NAME, IF NOT HEX ZERO
(9CC)	2508	BITSTRING	1	CPWALTTJ	NEW DESTINATION (RJE USER-ID)
(9CD)	2509	BITSTRING	1	CPWALTID	NEW SYSID
(9CE)	2510	BITSTRING	1	CPWALTF1	FLAG BYTE 1
		1...		CPWALTAD	"X'80'" ..ALTER DESTINATION (USER-ID)
		.1..		CPWALTAS	"X'40'" ..ALTER SYSID
		..1.		CPWALTAN	"X'20'" ..ALTER TARGET NODE NAME
		...1		CPWALTAU	"X'10'" ..ALTER TARGET USER ID
		.... 1...		CPWALTRD	"X'08'" ..RESET TEMP DISPOSITION
		.... .1..		CPWALTDT	"X'04'" ..RESET DUE TIME/INFO
(9CF)	2511	BITSTRING	1	CPWALTSG	NEW SEGMENT REQUEST
		11.. ..11		CPWALTSC	"C'C'" ..SEGMENT ON CARD BOUNDARY
		11.1 .111		CPWALTSP	"C'P'" ..SEGMENT ON PAGE BOUNDARY
		11.. 1..1		CPWALTSI	"C'I'" ..SEGMENT IMMEDIATELY
(9D0)	2512	CHAR-ACTER	4	CPWALTFN	NEW FORMS ID
(9D4)	2516	CHAR-ACTER	8	CPWALTDS	NEW DISTRIBUTION CODE
(9DC)	2524	CHAR-ACTER	16	CPWALTUI	NEW USER INFO
(9EC)	2540	BITSTRING	8		RESERVED FOR FUTURE USE
(9F4)	2548	BITSTRING	1	CPWAFL1	PALTER FLAG BYTE 1
		1...		CPWA1SEA	"X'80'" ..SEARCH OPERAND PRESENT
		VARIABLES USED BY POFFLOAD (SELECT) COMMAND PROCESSOR			
(AB8)	2744	SIGNED	4	CPWOFFCT	ADDR OF @SELECT ENTRY IN THE COMMAND CODE TABLE
		VARIABLES USED BY TLBL= AND LTAPE= KEYWORDS (BAM TAPE PROCESSING) <CLEARED BEFORE EACH COMMAND>			
(ABC)	2748		0	CPWLABBG	"*" WORKAREA BEGIN
(ABC)	2748	CHAR-ACTER	7	CPWLABNM	LABEL DTF NAME
(AC3)	2755	BITSTRING	1	CPWLABFG	FLAG BYTE
		1...		CPWTLBL	"X'80'" ..TLBL= SPECIFIED (BAM LABELED TAPE)
		.1..		CPWLTAPY	"X'40'" ..LTAPE=YES SPECIFIED (BAM LABELED TAPE)
		..1.		CPWLTAPN	"X'20'" ..LTAPE=NO SPECIFIED (BAM LABELED TAPE)
		...1		CPWOP5ON	"X'10'" ..5TH OPER.SPEC'D FOR L/P/R
(AC4)	2756		1	CPWOP#	KEYWORD COUNTER
		VARIABLES USED BY POFFLOAD JOURNAL=LST KEYWORD			
(AC5)	2757	BITSTRING	1	CPWOFJOU	JOURNAL=LST SPEC'D(X'FF')

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(AC6)	2758	BITSTRING	1	(3)	RESERVED
(ACC)	2764	ADDRESS	4	(15)	RESERVED
		.1.. 11..		CPWLABLN	"*-CPWLABBG" WORKAREA LEN
		WORK AREA USED BY COMMAND CONFIRMATION			
(B08)	2824	CHARACTER	9	CPWERPLA (0)	REPLY AREA FOR CONFIRMATION
(B08)	2824	CHARACTER	1	CPWERPLL	LENGTH OF REPLY AREA
(B09)	2825	CHARACTER	8	CPWERPLY	REPLY AREA
	ALIGN TO LINE BOUNDARY AND FILL UP WITH ZEROS M				
(B11)	2833	ADDRESS	1	(0)	
(B11)	2833		0	CPWALN	"*-CPWADS" LENGTH OF CP WORKAREA

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
				MAP OF COMMAND CODE TABLE ENTRY	
(0)	0	STRUCTURE	0	COMMAND	, DEFINE DUMMY SECTION
(0)	0	CHARACTER	7	COMLONG	LONG FORMAT OF COMMAND
(7)	7	CHARACTER	1	COMSHORT	SHORT FORMAT OF COMMAND
(8)	8	ADDRESS	4	COMADDR	ENTRY POINT ADDR OF COMMAND PROCESS
(C)	12	ADDRESS	1	COMMAXOP	MAX # OF OPERANDS ALLOWED PER CMND
(D)	13	CHARACTER	1	COMPFLG1	PERMISSION FLAG 1 (SHUTDOWN)
		111. 1...		COMYES	"C'Y'" .. INDICATES THAT COMMAND IS ALLOWED
		11.1 .1.1		COMNO	"C'N'" .. IND THAT COMMAND IS NOT ALLOWED
(E)	14	CHARACTER	1	COMPFLG2	PERMISSION FLAG 2 (AUTOSTART)
(F)	15	BITSTRING	1	COMLDFLG	LOAD FLAG (IND TO WHICH AREA COMMAND BELONGS).
		1...		COMLBASE	"X'80'" .. BASE COMMAND
		.1..		COMLRJE	"X'40'" .. RJE COMMAND (BSC/SNA)
		..1.		COMLSHR	"X'20'" .. SHARED COMMAND
		...1		COMLNET	"X'10'" .. NETWORKING COMMAND (PNET)
		.... 1...		COMLDEL	"X'08'" .. DISABLE SHORT CMD FORM
		.... .1..		COMLCONF	"X'04'" .. CONFIRM COMMAND
		.... ..1.		COMLCON1	"X'02'" .. CONFIRM PSTOP,PART .. CONFIRM PRELEASE,Q,ALL
		...1		CMNDLN	"*-COMMAND" LENGTH OF COMMAND TABLE ENTRY

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
				LAYOUT OF ONE OPERAND AFTER CONVERSION TO FIXED FORMAT	
(0)	0	STRUCTURE	0	OPERAND	, DEFINE DUMMY SECTION
	IF OPERAND LAYOUT CHANGES IPW$DLW MUST BE IN SYNC				
(0)	0	BITSTRING	1	OPLEN	LENGTH OF 'ORIGINAL' OR 'KEYOP'
(1)	1	BITSTRING	1	OPSWITCH	FLAG BYTE
		1...		OPSWHEX	"X'80'" ..INDICATES THAT 'OPHEX' CONTAINS THE BINARY REPRESENTATION OF THE OPERAND INTERPRETED AS HEXADECIMAL
		.1..		OPSWDEC	"X'40'" ..INDICATES THAT 'OPDEC' CONTAINS THE DECIMAL REPRESENTATION OF THE OPERAND INTERPRETED AS A DEC. TERM

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
		..1.		OPSWX	"X'20'" ..INDICATES THAT X'---' HAS BEEN STRIPPED OFF, SO DON'T TRY DECIMAL CONVERSION.
		...1		OPSWALFM	"X'10'" .. VALID ALPHAMERIC TERM FOUND
		.... 1...		OPSWKEY	"X'08'" .. KEYWORD OPERAND
		.... .1..		OPSWSTAR	"X'04'" .. '*' PRECEDED OPERAND
		.... ..1.		OPSWMIN	"X'02'" .. '-' PRECEDED OPERAND
		1		OPSWPLUS	"X'01'" .. '+' PRECEDED OPERAND
(2)	2	BITSTRING	1	OPFLAG2	FLAG BYTE 2
		1...		OPSWCAL	"X'80'" .. INDICATES THAT THE FIRST CHAR IS ALPHABETIC, REST IS ALPHAMERIC (A - Z, #, $, @, 0 - 9)
		.1..		OPSWQUO	"X'40'" .. OP WAS EMBEDDED BY QUOTES
		..1.		OPSWPAR	"X'20'" .. INDICATES THAT THE OPERAND WAS PRECEDED BY 'PARM=' AND EMBEDDED IN QUOTES
		...1		OPSWLXO	"X'10'" .. LAST PXMIT OPERAND
(3)	3	BITSTRING	1	OPMASKB	MASK BYTE USED FOR COMPARI'N
(4)	4	CHARACTER	24	ORIGINAL (0)	ORIGINAL OPERAND PADDED BLNK
(4)	4	CHARACTER	16	OPKEYOP	KEYWORD OPERAND PADDED BLANK
(14)	20	CHARACTER	8	OPKEYWRD	KEYWORD, PADDED WITH BLANKS
(1C)	28	BITSTRING	2	OPHEX	ZONED HEXADECIMAL INPUT CONVERTED TO BINARY
(1E)	30	BITSTRING	4	OPDEC	ZONED DECIMAL INPUT CONVERTED TO BINARY
		..1. ..1.		OPERLEN	"*-OPERAND" LENGTH OF FIXED FORMAT OPERAND AREA

VSE/POWER COMMAND CODE ABREVIATIONS

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
		11.1 ...1		PACCOUNT	"C'J'" VALID ABBRE. FOR PACCOUNT CMND
		11.. ...1		PALTER	"C'A'" VALID ABBRE. FOR PALTER CMND
		11.. ..1.		PBRDCST	"C'B'" VALID ABBRE. FOR PBRDCST CMND
		11.. ..11		PCANCEL	"C'C'" VALID ABBRE. FOR PCANCEL CMND
		111. 1...		PCOPY	"C'Y'" VALID ABBRE. FOR PCOPY
		11.1 ..11		PDELETE	"C'L' VALID ABBRE. FOR PDELETE CMND "
		11.. .1..		PDISPLAY	"C'D'" VALID ABBRE. FOR PDISPLAY CMND
		11.. .11.		PFLUSH	"C'F'" VALID ABBRE. FOR PFLUSH CMND
		11.. .111		PGOX	"C'G'" VALID ABBRE. FOR PGO CM
		11.. 1...		PHOLD	"C'H'" VALID ABBRE. FOR PHOLD CMND
		11.. 1..1		PINQUIRE	"C'I'" VALID ABBRE. FOR PINQUIRE CMND
		11.1 .1.1		PDRAIN	"C'N'" VALID ABBRE. FOR PDRAIN COMMAND
		11.1 .11.		POFFLOAD	"C'O'" VALID ABBRE. FOR POFFLOAD CMND
		11.1 1..1		PRELEASE	"C'R'" VALID ABBRE. FOR PRELEASE CMND
		111. ..11		PRESTART	"C'T'" VALID ABBRE. FOR PRESTART CMND
		11.1 .1..		PSEGMENT	"C'M'" VALID ABBRE. PSEGMENT
		111. .1..		PSETUP	"C'U'" VALID ABBRE. FOR PSETUP CMND
		111. ..1.		PSTART	"C'S'" VALID ABBRE. FOR PSTART CMND
		11.1 .111		PSTOP	"C'P'" VALID ABBRE. FOR PSTOP COMMAND
		111. .111		PXMIT	"C'X'" VALID ABBRE. FOR PXMIT COMMAND
		111. .1.1		PVARY	"C'V'" VALID ABBRE. PVARY
		....		PRESET	"X'00'" INTERNAL ABBREV. FOR PRESET COMMAND
		11		PLOAD	"X'03'" INTERNAL ABBREV. FOR PLOAD COMMAND
		.... .1..		PACT	"X'04'" INTERNAL ABBREV. FOR PACT COMMAND
		.... .1.1		@SELECT	"X'05'" INTERNAL ABBREV. FOR SIMULATED POFFLOAD 'SELECT' OPERATOR INP. TO HAVE CORRECTLY FORMATTED

GENERAL USED EQUATES

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
		.111 11.1		QUOTE	"X'7D'" USED TO TEST FOR QUOTE CHARACTER
		.111 111.		EQUAL	"C'='" USED TO TEST FOR EQUAL SIGN
		.11. 1.11		COMMA	"C','" USED TO TEST FOR COMMA SIGN
		.1..		BLANK	"C' '" USED TO TEST FOR BLANK CHARACTER
		.1.1 11..		STAR	"C'*'" USED TO TEST FOR ASTERIK SIGN
		.1.. 111.		PLUS	"C'+'" USED TO TEST FOR PLUS SIGN
		.11.		MINUS	"C'-'" USED TO TEST FOR MINUS SIGN
		.1.. 1.11		PERIOD	"C'.'" USED TO TEST FOR PERIOD

Communicator Information Block (CIB)

The communicator information block controls all access to the Notify message queue. It is created by the Spool access service master task at VSE/POWER initialization time.

It is addressed by field CACI in the CAT.

Definition Macro: IPW$DCI

Offset Hex	Type	Len	Name (Dim)	Description
	COMMUNICATOR INFORMATION BLOCK			
(0)	CHARACTER	16	CIBSD	SECTION DESCRIPTOR
(10)	DBL WORD	8	CIBWW	WORK AREA
(18)	SIGNED	4		RESERVED
(1C)	SIGNED	4	CIBLW	LOCKWORD
(20)	CHARACTER	48	CIBSV (0)	REGISTER SAVE AREA
(20)	SIGNED	4	CIBRE	REGISTER 14
(24)	SIGNED	4	CIBRF	REGISTER 15
(28)	SIGNED	4	CIBR0	REGISTER 0
(2C)	SIGNED	4	CIBR1	REGISTER 1
(30)	SIGNED	4	CIBR2	REGISTER 2
(34)	SIGNED	4	CIBR3	REGISTER 3
(38)	SIGNED	4	CIBR4	REGISTER 4
(3C)	SIGNED	4	CIBR5	REGISTER 5
(40)	SIGNED	4	CIBR6	REGISTER 6
(44)	SIGNED	4	CIBR7	REGISTER 7
(48)	SIGNED	4	CIBR8	REGISTER 8
(4C)	SIGNED	4	CIBR9	REGISTER 9
	GENERAL NOTIFY SECTION			
(50)	ADDRESS	4	CIBNTCB	ADDR OF NOTIFY TASK TCB
(54)	SIGNED	2	CIBMM#	MAX NUMBER OF MESSAGES IN QUEUE
(56)	SIGNED	2	CIBLMC	LOST MESSAGE COUNT
(58)	BITSTRING	1	CIBNACT	NOTIFY ACTION BYTE
	1...		CIBNASI	"X'80'" .. START SENDING TO ICCF
	.1..		CIBNASD	"X'40'" .. START SENDING TO DSNX
	..1.		CIBNACM	"X'20'" .. MSG ADDED FOR SUBSYSTEM
	...1		CIBNANS	"X'10'" .. SUBSYS CONNECTION ESTABL
(59)	BITSTRING	3		RESERVED
(5C)	ADDRESS	4	CIBFCIE	1ST ENTRY IN CIE-CHAIN
	VSE/ICCF SUBSYSTEM SECTION			
(60)	ADDRESS	4	CIBIMSG	ADDRESS OF 1ST MESSAGE IN QUEUE
(64)	ADDRESS	4	CIBIMTL	*- TAIL POINTER (MSG QUEUE)
(68)	ADDRESS	4	CIBIMBS	ADDR OF MESSAGE BEING SENT
(6C)	ADDRESS	4	CIBICM#	CURRENT NO. OF MESSAGES IN QUEUE
(70)	BITSTRING	1	CIBIFLG	FLAG BYTE 1
	1...		CIBISIP	"X'80'" .. SEND IN PROGRESS
	.1..		CIBICON	"X'40'" .. CONNECTION COMPLETED
	1		CIBIQNE	"X'01'" .. MESSAGE ADDED TO QUEUE
(71)	BITSTRING	3		RESERVED FOR FUTURE USE
(74)	ADDRESS	4	CIBIXPCC	ADDR OF USED XPCCB
	VSE/DSNX SUBSYSTEM SECTION			
(78)	ADDRESS	4	CIBDMSG	ADDR 1ST MSG REC IN QUEUE
(7C)	ADDRESS	4		ADDR LAST MSG REC IN QUEUE
(80)	ADDRESS	4	CIBDMBS	ADDR OF MSG REC BEING SENT
(84)	BITSTRING	1	CIBDFLG	FLAG BYTE 1
	1...		CIBDSIP	"X'80'" .. SEND IN PROGRESS
	.1..		CIBDCON	"X'40'" .. CONNECTION COMPLETED

Offset Hex	Type	Len	Name (Dim)	Description
	1		CIBDQNE	"X'01'" .. MESSAGE ADDED TO QUEUE
(85)	BITSTRING	3		RESERVED FOR FUTURE USE
(88)	ADDRESS	4	CIBDXPCC	ADDRESS OF USED XPCCB
	CROSS PARTITION SUPPORT SECTION			
(8C)	BITSTRING	1	CIBACT	ACTION BYTE
	1...		CIBASTP	"X'80'" .. TERMINATE MASTER TASK
(8D)	BITSTRING	1	CIBMSTAT	STATUS BYTE
	1...		CIBMINT	"X'80'" .. MASTER TASK INITIALIZED
	.1..		CIBMXPE	"X'40'" .. WRONG XPCCB
	..1.		CIBMTERM	"X'20'" .. TERMQUIESCE ISSUED
(8E)	BITSTRING	2	CIBUTOK	TOKEN OF X-PART. USER TASK
(90)	ADDRESS	4	CIBMXPT	ADDR OF X-PART. MASTER TCB
(94)	ADDRESS	4	CIBMXPCC	ADDR OF CONNECT ANY XPCCB
(98)	CHARACTER	8	CIBMXIDK	IDENTIFY TOKEN
(A0)	CHARACTER	8	CIBMXIDD	IDENTIFY TOKEN OF DST POWER
	VSE/OCCF HEARTBEAT SECTION			
(A8)	ADDRESS	4	CIBHTCB	ADDR OF HEARTBEAT TASK
(AC)	ADDRESS	4		RESERVED
	1.11		CIBLN	"*-CIBDS" LENGTH OF CONTROL BLOCK

Class Table Entry

When a queue entry is added to the master record class table, it is chained to some class table entry described here, contained in either the RDR, LST, PUN or XMT queues which make up the class table.

Definition Macro: IPW$DCT

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
					CLASS TABLE ENTRY
(0)	0	STRUCTURE	0	CTDS	, DUMMY SECTION DEFINITION
(0)	0	SIGNED	4	CTQF	FIRST IN CLASS ADDRESS
(4)	4	SIGNED	4	CTQL	LAST IN CLASS ADDRESS
		.... 1...		CTLN	"*-CTDS" LENGTH DESCRIPTOR

Communicator Information Block 2 (CI2)

Definition Macro: IPW$DCI

The Communicator Information Block 2 (CI2) controls all access to the job completion message queue. It contains the address of the first ACIE entry of the ACIE queue. It is created at initialization time by the cross partition master task. The CI2 is addressed by an address in the permanent area (CAT, field CACI2).

Offset Hex	Type	Len	Name (Dim)	Description
THE COMMUNICATOR INFORMATION BLOCK 2 (CIB2)				
(0)	CHARACTER	16	CI2SD	SECTION DESCRIPTOR
(10)	DBL WORD	8	CI2WW	WORK AREA
(18)	SIGNED	4		RESERVED
(1C)	SIGNED	4	CI2LW	LOCKWORD
(20)	CHARACTER	48	CI2SV (0)	REGISTER SAVE AREA
(20)	SIGNED	4	CI2RE	REGISTER 14
(24)	SIGNED	4	CI2RF	REGISTER 15
(28)	SIGNED	4	CI2R0	REGISTER 0
(2C)	SIGNED	4	CI2R1	REGISTER 1
(30)	SIGNED	4	CI2R2	REGISTER 2
(34)	SIGNED	4	CI2R3	REGISTER 3
(38)	SIGNED	4	CI2R4	REGISTER 4
(3C)	SIGNED	4	CI2R5	REGISTER 5
(40)	SIGNED	4	CI2R6	REGISTER 6
(44)	SIGNED	4	CI2R7	REGISTER 7
(48)	SIGNED	4	CI2R8	REGISTER 8
(4C)	SIGNED	4	CI2R9	REGISTER 9
(50)	SIGNED	2	CI2MM#	MAX. # OF POSSIBLE JCM MSGS
(52)	SIGNED	2	CI2LMC	MAX. # OF LOST JCM MSGS
(54)	BITSTRING	4		RESERVED
(58)	ADDRESS	4	CI2FCIE	ADDRESS OF BCA OF 1ST ACIE
(5C)	ADDRESS	4	CI2LCIE	ADDRESS OF BCA OF LAST ACIE
(60)	ADDRESS	4	CI2CIE	ADDRESS OF 1ST ACIE
(64)	BITSTRING	24	CI2TQE	TIMER QUEUE ELEMENT
(7C)	BITSTRING	8		CRITICAL XPCC APPLICATION
(84)	BITSTRING	8		CRITICAL SAS USER
	1... 11..		CI2LN	"*-CI2DS" LENGTH OF CONTROL BLOCK

Communicator Information Element (CIE)

The communicator information element is built by the spool access service master task when a 'notify' communication path is established with a VSE/AF subsystem, such as CICS/VS. One CIE exists for each established 'notify' communication path. The CIEs are chained off the CIB.

Definition Macro: IPW$DCI

Offset Hex	Type	Len	Name (Dim)	Description
	COMMUNICATOR INFORMATION ELEMENT			
(0)	CHARACTER	16	CIESD	SECTION DESCRIPTOR
(10)	CHARACTER	8	CIEAPPL	SUBSYSTEM NAME
(18)	ADDRESS	4	CIENEXT	ADDRESS OF NEXT CIE OR 0
(1C)	ADDRESS	4	CIEXPCC	ADDRESS OF XPCCB
(20)	ADDRESS	4	CIEHEAD	ADDRESS OF 1ST MESSAGE
(24)	ADDRESS	4	CIETAIL	ADDRESS OF LAST MESSAGE
(28)	ADDRESS	4	CIEAMBS	ADDR OF MESSAGE BEING SENT
(2C)	BITSTRING	1	CIESTAT	STATUS BYTE
	1...		CIESIPR	"X'80'" .. SEND IN PROGRESS
	.1..		CIEEADD	"X'40'" .. MESSAGE ADDED TO QUEUE
	..1.		CIEECBL	"X'20'" .. ECB WAIT LIST UPDATED
(2D)	BITSTRING	1		RESERVED
(2E)	SIGNED	2		RESERVED
	..11		CIELEN	"*-CIEDS" LENGTH OF CONTROL BLOCK

Application Communicator Information Element (ACIE)

The Application Communicator Information Element is built by

- Execution reader task
- Network receiver task
- Timer task

when a fixed format job completion message contained in a nodal message record (NMR) reached its final destination for later retrieval by an application program. Such an ACIE is created for each pair of XPCC application-ID and Spool-Access support user-id contained in the NMR.

Definition Macro: IPW$DCI

Offset Hex	Type	Len	Name (Dim)	Description
COMMUNICATOR INFORMATION ELEMENT FOR JCM RETRIEVAL (ACIE)				
(0)	CHARACTER	16	ACIESD	SECTION DESCRIPTOR
(10)	CHARACTER	8	ACIEAPPL	XPCC-APPLICATION IDENTIFIER
(18)	CHARACTER	8	ACIEUSER	USER-ID
(20)	ADDRESS	4	ACIENEXT	ADDRESS OF NEXT ACIE
(24)	SIGNED	4		RESERVED FOR FUTURE USE
(28)	ADDRESS	4	ACIEHEAD	ADDR. OF BCA OF 1ST MSG
(2C)	ADDRESS	4	ACIETAIL	ADDR. OF BCA OF LAST MSG
(30)	ADDRESS	4	ACIECMSG	ADDR. OF BCA OF CUR'NT MSG
(34)	SIGNED	4		RESERVED FOR FUTURE USE
(38)	SIGNED	2	ACIELMC	# OF LOST MSG IN THIS QUEUE
(3A)	SIGNED	2	ACIECM#	CURRENT MESSAGE NUMBER
(3C)	SIGNED	4		RESERVED FOR FUTURE USE
	.1..		ACIELN	"*-ACIEDS" LENGTH OF CONTROL BLOCK

Control Address Table (CAT)

Definition macro: IPW$DPA

The control address table consists of a set of tables, addresses, and constants, used to link the component routines of the VSE/POWER subsystem during execution. The control address table is located in the SVA part fixed by VSE/POWER. A pointer to it can be found at offset X'14' from the VSE/POWER partition (partition save area register 10) or in field IJBPWR of the system communication region. Its format as it is printed in a dump is shown below.

Register 10 always points to the beginning of the CAT. The fields in the CAT may be found by using register 10 as base register.

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
CONTROL ADDRESS TABLE (CAT)					
THE FOLLOWING DUMMY SECTION DESCRIBES THE FORMAT AND ORGANIZATION OF THE CONTROL ADDRESS TABLE WHICH RESIDES IN THE SVA AS THE FIRST PART OF THE NUCLEUS. FIELDS USED BY VSE/AF ARE FLAGGED WITH '(AF)'.					
(0)	0	STRUCTURE		PADS	(AF) CONTROL ADDRESS TABLE
(0)	0	CHARACTER	16	PASD	STORAGE DESCRIPTOR
(10)	16	CHARACTER	37	PACOPY	
(35)	53	CHARACTER	7		
		..1. 11..		PACOPYL	"*-PACOPY"
(3C)	60	CHARACTER	4		RES FOR FUTURE LAYOUT
THE FOLLOWING EVENT CONTROL BLOCK IS USED TO SIGNAL TO VSE/POWER THAT IT HAS WORK TO DO.					
(40)	64	SIGNED	4	PAEB	(AF) VSE/POWER MASTER ECB
THE FOLLOWING FIELDS DEFINE THE KEY BOUNDARIES WITHIN THE VIRTUAL ADDRESS SPACE OF THE VSE/POWER PARTITION, DIVIDING IT INTO THE PERMANENT AREA, THE FIXABLE AREA, AND THE PAGEABLE AREA.					
(44)	68	ADDRESS	4	PAPA	START OF VSE/POWER PARTITION
(48)	72	ADDRESS	4	PAFA	START OF FIXABLE AREA
(4C)	76	ADDRESS	4	PAVA	START OF PAGEABLE AREA
(50)	80	ADDRESS	4	PAEN	END OF VSE/POWER PARTITION
THE FOLLOWING FIELDS DEFINE THE BOUNDARIES OF THE SYSTEM LOGICAL TRANSIENT AREA.					
(54)	84	ADDRESS	4	PALS	START OF LTA
(58)	88	ADDRESS	4	PALE	END OF LTA +1
THE FOLLOWING FIELDS CONTAIN THE BEGIN/END ADDRESS OF THE SYSTEM GETVIS AREA AND THE SHARED VIRTUAL AREA. THESE FIELDS ARE USED BY THE NUCLEUS DATA AREA VALIDATION ROUTINE.					
(5C)	92	ADDRESS	4	PAESVA	ADDR LAST BYTE SVA
(60)	96	ADDRESS	4	PASVA	ADDR FIRST BYTE SVA
THE FOLLOWING FIELDS CONTAIN THE ALET OF THE POWER PARTITION, THE ADDRESS OF THE PCE AND THE TIK OF THE VSE/POWER PARTITION. THESE FIELDS ARE USED BY THE APPENDAGE ROUTINES TO ACTIVATE THE VSE/POWER PARTITION WHEN APPROPRIATE.					

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(64)	100	SIGNED	4	CAPALET	ALET OF POWER PARTITION
(68)	104	ADDRESS	4	CAPCE	(AF) ADDRESS OF POWER PCE
(6C)	108	SIGNED	2	CATI	(AF) VSE/POWER TASK ID
(6E)	110	SIGNED	2	CATS	TASK ID OF TIMER SUBTASK

EXTERNAL INTERFACE ADDRESSES
THE FOLLOWING ADDRESS CONSTANTS REPRESENT THE ADDRESSES
OF APPENDAGE ROUTINES LOCATED WITHIN THE VSE/POWER
NUCLEUS AND ARE USED TO ESTABLISH ENTRY TO THE ROUTINES
FROM THE DOS/VSE SUPERVISOR AND THE ATTENTION ROUTINE
AND THE JCL/VSE.

(70)	112	ADDRESS	4	CAAI	(AF) ATTENTION INTERFACE
(74)	116	ADDRESS	4	CAPF	PAGE FAULT APPENDAGE
(78)	120	ADDRESS	4	CAHR	(AF) HOT READER ROUTINE
(7C)	124	ADDRESS	4	CACE	RJE CE APPENDAGE
(80)	128	ADDRESS	4	CA00	(AF) SVC 0 ROUTINE
(84)	132	ADDRESS	4	CA90	(AF) SVC90/SVC91 APPENDAGE
(88)	136	ADDRESS	4	CAEOJ	(AF) END OF JOB EXIT ROUTINE
(8C)	140	ADDRESS	4	CASEGMI	IPWSEGM INTERFACE ROUTINE

$MXSEGMI EQU X'8C' CAT OFFSET OF 'CASEGMI' ...
AS DEFINED IN IPW$MXD, ..
AND USED IN IPWSEGM !!!!!

(90)	144	ADDRESS	4	CAFTTR00	FULL TASK TRACE ROUTINE
(94)	148	ADDRESS	4	(2)	UNUSED

ADDRESS OF EXTERNALLY SPECIFIED GENERATION TABLE

(9C)	156	ADDRESS	4	CAGEN	ADDRESS OF POWER GENER. TBLE

EXTERNAL ADDRESS(ES) FOR RJE,BSC

(A0)	160	ADDRESS	4	CABM	BSC FUNCTION ENTRY POINT

CROSS-PARTITION CONTROL INFORMATION
THE FOLLOWING ARE USED BY SPOOL MANAGEMENT IN MAINTAINING
CROSS-PARTITION XECB INFORMATION.
IN-CORE READER CROSS-PARTITION XECB

(A4)	164	SIGNED	4	ICXP	IN-CORE READER XECB
(A8)	168	BITSTRING	1		UNUSED
(A9)	169	ADDRESS	3	ICTA	XECBTAB ADDR OF ICR XECB

SPOOL/COMMAND MANAGER CROSS-PARTITION XECB

(AC)	172	SIGNED	4	SMXP	SPOOL/COMMAND MGR XECB
(B0)	176	BITSTRING	1		UNUSED
(B1)	177	ADDRESS	3	SMTA	XECBTAB ADDR OF SPM XECB

THE FOLLOWING TWO FIELDS CONTAIN THE PIK OF SPOOL/COMMAND
MANAGER AS WELL AS WELL AS THE PIK OF THE INCORE READER
IF APPLICABLE.

(B4)	180	SIGNED	2	IPIK	IN-CORE RDR USER'S PIK
(B6)	182	SIGNED	2	SPIK	SPOOL/COMMAND MGR USER'S PIK

XECB WAITM LIST USED BY SPOOL MANAGER MASTER TCB FOR SELECTING
THE IN-CORE RDR TASK AND/OR THE SPOOL/COMMAND MANAGER TASK.

(B8)	184	ADDRESS	4	ICWL	ADDR(ICXP)
(BC)	188	ADDRESS	4	SPWL	ADDR(SMXP)
(C0)	192	BITSTRING	1		WAITM LIST DELIMITER

WAIT TIME FOR OPNDST IN MINUTES USED BY IPW$$S2

(C1)	193	BITSTRING	1	CAOPNDST	OPNDST TIME INTERVAL IN MIN
(C2)	194	BITSTRING	1		UNUSED

MISCELLANEOUS

(C3)	195	ADDRESS	1		UNUSED
(C4)	196	SIGNED	4	CAUNBECB	(AF) ECB, POSTED IF PART UNBATCH
(C8)	200	SIGNED	4	CAUNBCTS	NO OF PART. UNBATCH ISSUED

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(CC)	204	SIGNED	4	CAUNBCTP	(AF) NO OF PART. UNBATCH PROCES.
(D0)	208	CHARACTER	2	CAAPID	(AF) ACTIVE PARTITION ID (PID)
(D2)	210	SIGNED	2		UNUSED
COMMAND PROCESSOR:					
(D4)	212	ADDRESS	4	CACMTBL	ADDRESS OF COMMAND TABLE
(D8)	216	ADDRESS	4	CAARECB	ATTENTION ROUTINE ECB
(DC)	220	ADDRESS	4	CACMT@S	ADDR POFFLOAD DUMMY ENTRY
(E0)	224	SIGNED	4		UNUSED
TIMER SERVICE CONTROL INFORMATION					
(E4)	228	BITSTRING	1	CATF	FLAG BYTE
		1...		CATV	"X'80'" .. TIMER EXPIRED
		.1..		CATF40	"X'40'" .. ***** UNUSED *****
		..1.		CATMSOFF	"X'20'" .. SUPPRESS 1QZ2A
		...1		CATF10	"X'10'" .. ***** UNUSED *****
		.... 1...		CATDLOW	"X'08'" .. USE LOW DPST PRIORITY
		.... .1..		CATFCKCW	"X'04'" .. ISSUE 1R30I AND CANCEL
		.... ..1.		CATFNSLI	"X'02'" .. RESTRICT SLI FOR 'FROM'
		1		CATFALBD	"X'01'" .. PALTER CHANGES BOTH DISP
(E5)	229	BITSTRING	1	CATF2	FLAG BYTE 2
		1...		CATF2ISP	"X'80'" .. USE ISEP FOR LST TASKS
		.1..		CATF2NC9	"X'40'" .. NO CHANNEL 9/12 POSTING
		..1.		CATF2DLS	"X'20'" .. USE DLSEP FOR LST TASKS
		...1		CATF2SPF	"X'10'" .. FORCE SEPARATOR PAGE FOR SET DLSEP AND SET ISEP
		.... 1...		CATFSISP	"X'08'" .. USE ISEPSAS FOR SAS TASK
		.... .1..		CATFSDLS	"X'04'" .. USE DLSEPSAS FOR SAS TASK
		.... ..1.		CATFSSPF	"X'02'" .. FORCE SEPARATOR PAGE FOR SET ISEPSAS AND DLSEPSAS
(E6)	230	BITSTRING	2	CATF3	FLAG BYTE 3
		1...		CATF3UIP	"X'80'" .. UPDATE IN PROGRESS FOR TCB OR BCW CHAIN
(E7)	231	BITSTRING	1		UNUSED
(E8)	232	ADDRESS	4	CATQ	ADDR OF FIRST TQE IN CHAIN
(EC)	236	ADDRESS	4	CAIT	TIMER INTERVAL EXIT ROUTINE
(F0)	240	SIGNED	4	CASE	TIMER DOS/VSE SUBTASK ECB
(F4)	244	ADDRESS	4		UNUSED
(F8)	248	ADDRESS	4		UNUSED
(FC)	252	ADDRESS	4		UNUSED
RESOURCE LOCKWORD TABLE IN CASE OF TASK TERMINATION THE TERMINATOR (IPW$$TR) WILL SCAN THE RESOURCE LOCKWORD TABLE					
(100)	256	ADDRESS	4	CAFT	FCB TABLE LOCKWORD
(104)	260	ADDRESS	4	CBLW	RJE BSC LOCKWORD
		.... ..1.		CA#L	"(*-CAFT)/4" # OF LOCKWORDS
(108)	264	ADDRESS	4		UNUSED
RESOURCE CONTROL BLOCK ADDRESSES THE FOLLOWING ADDRESS CONSTANTS REPRESENT THE ADDRESSES OF MAJOR VSE/POWER CONTROL BLOCKS. EACH BLOCK IS FORMATTED AS A RESOURCE CONTROL BLOCK AND CONTAINS A LOCKWORD IN ITS EIGHTH FULL WORD.					
(10C)	268	ADDRESS	4	CAQC	DISK MANAGEMENT BLOCK
(110)	272	ADDRESS	4	CAAC	ACCOUNT CONTROL BLOCK
(114)	276	ADDRESS	4	CASC	STORAGE CONTROL BLOCK
(118)	280	ADDRESS	4	CAMM	MESSAGE CONTROL BLK - LOCAL
(11C)	284	ADDRESS	4	CARM	MESSAGE CONTROL BLK - RMOTE
(120)	288	ADDRESS	4	CASM	SNA CONTROL BLOCK ADDRESS
(124)	292	ADDRESS	4	CAGP	GENERAL PURPOSE WORK AREA
(128)	296	ADDRESS	4	CAAB	ASYN SERVICE ANCHOR BLOCK
(12C)	300	ADDRESS	4	CATK	PTR TRACE INFORMATION BLOCK
(130)	304	ADDRESS	4	CAPN	PTR TO PNET CONTROL BLOCK

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(134)	308	ADDRESS	4	CACI	PTR COMMUNICATOR INF BLOCK
(138)	312	ADDRESS	4	CAEDCB	MASTER EXTERNAL DEVICE CB
(13C)	316	ADDRESS	4	CADPCB	DYNAMIC PARTITION CTL BLOCK
(140)	320	ADDRESS	4	CAVS	VIRTUAL STORAGE CONTROL BLCK
(144)	324	ADDRESS	4	CACI2	PTR TO 2ND COMM. INF. BLOCK
(148)	328	ADDRESS	4		UNUSED

MODULE CONTROL BLOCK (MCB) ADDRESS TABLE
THE FOLLOWING ADDRESS CONSTANTS REPRESENT THE ADDRESSES OF
THE MODULE CONTROL BLOCKS WHICH ARE ASSOCIATED WITH EACH
EXTENT OF DISK STORAGE SPECIFIED TO THE VSE/POWER SYSTEM
AS INTERMEDIATE STORAGE.
MODULE CONTROL BLOCKS ARE RESOURCE CONTROL BLOCKS AND
PROCESSED AS SUCH BY VSE/POWER.

(14C)	332	ADDRESS	4	CAA0	RESERVED
(150)	336	ADDRESS	4	CAQ1	MCB QUEUE FILE
(154)	340	ADDRESS	4	CAD2	MCB DATA FILE - MODULE 1
(158)	344	ADDRESS	4	CAD3	MCB DATA FILE - MODULE 2
(15C)	348	ADDRESS	4	CAD4	MCB DATA FILE - MODULE 3
(160)	352	ADDRESS	4	CAD5	MCB DATA FILE - MODULE 4
(164)	356	ADDRESS	4	CAD6	MCB DATA FILE - MODULE 5
(168)	360	ADDRESS	4	CAD7	MCB DATA FILE - MODULE 6
(16C)	364	ADDRESS	4	CAD8	MCB DATA FILE - MODULE 7
(170)	368	ADDRESS	4	CAD9	MCB DATA FILE - MODULE 8
(174)	372	ADDRESS	4	CAD10	MCB DATA FILE - MODULE 9
(178)	376	ADDRESS	4	CAD11	MCB DATA FILE - MODULE 10
(17C)	380	ADDRESS	4	CAD12	MCB DATA FILE - MODULE 11
(180)	384	ADDRESS	4	CAD13	MCB DATA FILE - MODULE 12
(184)	388	ADDRESS	4	CAD14	MCB DATA FILE - MODULE 13
(188)	392	ADDRESS	4	CAD15	MCB DATA FILE - MODULE 14
(18C)	396	ADDRESS	4	CAD16	MCB DATA FILE - MODULE 15
(190)	400	ADDRESS	4	CAD17	MCB DATA FILE - MODULE 16
(194)	404	ADDRESS	4	CAD18	MCB DATA FILE - MODULE 17
(198)	408	ADDRESS	4	CAD19	MCB DATA FILE - MODULE 18
(19C)	412	ADDRESS	4	CAD20	MCB DATA FILE - MODULE 19
(1A0)	416	ADDRESS	4	CAD21	MCB DATA FILE - MODULE 20
(1A4)	420	ADDRESS	4	CAD22	MCB DATA FILE - MODULE 21
(1A8)	424	ADDRESS	4	CAD23	MCB DATA FILE - MODULE 22
(1AC)	428	ADDRESS	4	CAD24	MCB DATA FILE - MODULE 23
(1B0)	432	ADDRESS	4	CAD25	MCB DATA FILE - MODULE 24
(1B4)	436	ADDRESS	4	CAD26	MCB DATA FILE - MODULE 25
(1B8)	440	ADDRESS	4	CAD27	MCB DATA FILE - MODULE 26
(1BC)	444	ADDRESS	4	CAD28	MCB DATA FILE - MODULE 27
(1C0)	448	ADDRESS	4	CAD29	MCB DATA FILE - MODULE 28
(1C4)	452	ADDRESS	4	CAD30	MCB DATA FILE - MODULE 29
(1C8)	456	ADDRESS	4	CAD31	MCB DATA FILE - MODULE 30
(1CC)	460	ADDRESS	4	CAD32	MCB DATA FILE - MODULE 31
(1D0)	464	ADDRESS	4	CAD33	MCB DATA FILE - MODULE 32
		..1.		CA#DFE	"(*-CAD2)/4" NUMBER OF DATA FILE EXTENTS
		..11 ..1.		CA#R	"(*-CAQC)/4" NUMBER OF RESOURCES
(1D4)	468	ADDRESS	4		END OF LIST IDENTIFIER
(1D8)	472	ADDRESS	4		UNUSED

THE ABOVE VALUE IS USED BY THE TASK TERMINATOR ROUTINE
TO DETERMINE THE NUMBER OF RESOURCES TO BE SCANNED.
TASK STATE VALUES
THE FOLLOWING CONSTANTS ARE USED BY THE TASK MANAGEMENT
MACRO INSTRUCTIONS TO SET VALUES WITHIN THE TASK SELECTION
FIELDS OF THE TASK CONTROL BLOCK CORRESPONDING TO THE
INDIVIDUAL TASK STATES THAT THEY IDENTIFY.

(1DC)	476	CHARACTER	1	TMCI	THE TASK IS INACTIVE
(1DD)	477	ADDRESS	3		DO NOT SELECT THE TASK
(1E0)	480	CHARACTER	1	TMCP	PAGE FAULT IN PROCESS

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(1E1)	481	ADDRESS	3		DO NOT SELECT THE TASK
(1E4)	484	CHARACTER	1	TMCO	WAIT FOR OPERATOR
(1E5)	485	ADDRESS	3		DO NOT SELECT THE TASK
(1E8)	488	CHARACTER	53	TMCL	
(1E9)	489	ADDRESS	3		TEST LOCKWORD
(1EC)	492	CHARACTER	1	TMCM	WAIT ON MULTIPLE POSTING
(1ED)	493	ADDRESS	3		TEST CONTROL BLOCKS
(1F0)	496	CHARACTER	1	TMCQ	WAIT ON CLASS TABLE POSTING
(1F1)	497	ADDRESS	3		TEST CONTROL BLOCKS
(1F4)	500	CHARACTER	1	TMCC	WAIT ON SINGLE POSTING
(1F5)	501	ADDRESS	3		TEST CONTROL BLOCK, CHECK I/O TOO
(1F8)	504	CHARACTER	1	TMCS	WAIT ON SPACE POSTING
(1F9)	505	ADDRESS	3		TEST CONTROL BLOCK, CHECK I/O TOO
(1FC)	508	CHARACTER	1	TMCD	IMMEDIATE DISPATCH
(1FD)	509	ADDRESS	3		DISPATCH THE TASK
(200)	512	CHARACTER	1	TMCW	WAIT STATE
(201)	513	ADDRESS	3		WAIT ROUTINE
(204)	516	CHARACTER	1	TMCR	THE TASK IS RUNNING
(205)	517	ADDRESS	3		RE-SELECTION ADDRESS
(208)	520	CHARACTER	1	TMCB	WAIT ON RJE,BSC - PNET EVENT
(209)	521	ADDRESS	3		TEST TECB FOR RJE EVENT
(20C)	524	CHARACTER	1	TMCE	WAIT ON SINGLE ECB POSTING
(20D)	525	ADDRESS	3		TEST ECB POST BIT
(210)	524	CHARACTER	1	TMCX	WAIT ON MIXED POSTING
(211)	525	ADDRESS	3		TEST ECB POST BIT

PERMANENT TASK CONTROL BLOCKS

THE FOLLOWING TABLE CONTAINS THE ADDRESSES OF THE VSE/POWER TASK CONTROL BLOCKS WHICH ARE PERMANENTLY PRESENT IN FIXED STORAGE.

THE FIRST PART OF THE TABLE IS USED BY THE INITIATOR ROUTINE IPW$$I2 TO ESTABLISH THE INITIAL TASK LIST.

(214)	532	ADDRESS	4	TATM	WAIT CONTROL BLOCK
(218)	536	CHARACTER	4	TAOC	AR ROUTINE COMMAND PROC TCB
(21C)	540	CHARACTER	4	TAIT	INITIALIZATION/TERMINATION
(220)	544	ADDRESS	4	TALM	LINE MANAGER
(224)	548	ADDRESS	4	TASP	SPOOL MANAGER
(228)	552	ADDRESS	4	TATES	TIME EVENT SCHEDULING TASK
(22C)	556	ADDRESS	4	TADPST	DYNAMIC PART. SCHED. TASK
(230)	560	ADDRESS	4		UNUSED
(234)	564	ADDRESS	4		UNUSED

TASK CONTROL ADDRESS TABLE
THE FOLLOWING TABLE IS USED BY THE TASK INITIATION AND
TERMINATION ROUTINES TO DETERMINE THE POSITION IN THE TASK
LIST AT WHICH A NEW TASK OF A GIVEN TYPE IS TO BE INSERTED.
THE FIRST BYTE OF EACH ENTRY CONTAINS AN ALPHAMERIC
CHARACTER IDENTIFYING THE TYPE OF TASK TO WHICH THE ENTRY
RELATES. THE REMAINING THREE BYTES CONTAIN THE ADDRESS OF
THE TASK CONTROL BLOCK FOR THE MOST RECENTLY-ATTACHED TASK
OF THAT TYPE. IF NO SUCH TASK EXISTS THE ADDRESS CONTAINED
IN THE ENTRY IS THAT OF THE TASK CONTROL BLOCK OF THE
CURRENTLY-ATTACHED TASK WHICH MUST PRECEDE ANY NEW TASK OF
THE DESIGNATED TYPE.
NOTE, THAT THE FOLLOWING FIELDS: CAOP,CARJ,CAMP,CARW,CAEX,
AND CARR GET THE ADDRESS OF THE COMMAND PROCESSOR TCB
AT INITIALIZATION (IPW$$I2) TIME.
THE ATTACH SCHEME BELOW MAY BE OVERRULED AS FOLLOWS:
 1) WRITER TASK ('W') STARTED WITH OPTION 'VM'/'SP'
    IS CHAINED WITH LOWER PRIORITY THAN EXEC. PROC'S
    AMONST THE READER ('R') TASKS.
 2) WRITER TASK ('W') STARTED WITH OPTION 'HP'
    IS CHAINED WITH HIGHER PRIORITY AMONGST THE
    X-PARTITION SAS TASKS.
 3) DYN. PART. SCHED. TASK ('D') WITH AUTOSTART
          SET DYNAL=LOW
    IS CHAINED WITH LOWER PRIORITY THAN EXISTING
    STATIC AND DYNAMIC PARTITIONS AS THE FIRST
    READER ('R') TASK.

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(238)	568	CHAR-ACTER	1	CATRT	TASK IDENTIFYING PREFIX
(239)	569	ADDRESS	3		ADDR OF TRACE MONITOR TASK
(23C)	572	CHAR-ACTER	53	CALM	
(23D)	573	ADDRESS	3		ADDRESS OF LINE OR SNA MANAGER
(240)	576	CHAR-ACTER	1		TASK IDENTIFYING PREFIX
(241)	577	ADDRESS	3		ADDR OF TIMER TASK
(244)	580	CHAR-ACTER	1		TASK IDENTIFYING PREFIX
(245)	581	ADDRESS	3		ADDR OF TIME EVENT SCHEDULER
(248)	584	CHAR-ACTER	1		TASK IDENTIFYING PREFIX
(249)	585	ADDRESS	3		ADDRESS OF XPART OR DST TASK
(24C)	588	CHAR-ACTER	1	CASP	TASK IDENTIFYING PREFIX
(24D)	589	ADDRESS	3		ADDRESS OF SPOOL MANAGER
(250)	592	CHAR-ACTER	1	CAOP	TASK IDENTIFYING PREFIX
(251)	593	ADDRESS	3		ADDR. OF CMD PROCESSOR TCB
(254)	596	CHAR-ACTER	1	CARJ	TASK IDENTIFYING PREFIX
(255)	597	ADDRESS	3		ADDRESS OF LAST RJE TASK
(258)	600	CHAR-ACTER	1	CAMP	TASK IDENTIFYING PREFIX
(259)	601	ADDRESS	3		ADDRESS OF DYN. PART. SCHED.
(25C)	604	CHAR-ACTER	1	CARW	TASK IDENTIFYING PREFIX
(25D)	605	ADDRESS	3		ADDRESS OF LAST WRITER TASK
(260)	608	CHAR-ACTER	1	CAEX	TASK IDENTIFYING PREFIX
(261)	609	ADDRESS	3		ADDRESS OF LAST EXP TASK
(264)	612	CHAR-ACTER	1	CARR	TASK IDENTIFYING PREFIX
(265)	613	ADDRESS	3		ADDRESS OF LAST READER TASK, OR WRITER TASK STARTED WITH THE 'VM' OR 'SP' OPTION

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(268)	616	BITSTRING	1	CALD	END OF INDEX TABLE
(26C)	620	ADDRESS	4		UNUSED
	VSE/POWER FLAG BYTES				
(270)	624	BITSTRING	1	CAFLG1	FLAG BYTE 1
		1...		CAFHLD	"X'80'" ..PLACE PARTITIONS IN PAUSE
		.1..		CAFOFFB	"X'40'" ..POFFLOAD BACKUP RUNNING
		..1.		CAFTTR	"X'20'" ..TASK TRACE ENABLED
		...1		CAFHLDX	"X'10'" ..PAUSE PARTITION AT DISP=X
		.... 1...		CAFLOX	"X'08'" ..OUTPUT EXIT ENABLED
		.... .1..		CAFLRX	"X'04'" ..JOB EXIT ENABLED
		.... ..1.		CAFLPX	"X'02'" ..PNET EXIT ENABLED
		1		CAFLAB	"X'01'" ..ABN TERMINATION IN PROC.
(271)	625	BITSTRING	1	CAFLG2	FLAG BYTE 2
		1...		CAF2PIM	"X'80'" ..PEND IMM ISSUED
		.1..		CAF2NOP	"X'40'" ..DO NOT POST TERMINATION
		..1.		CAF2RIP	"X'20'" ..ISSUE RE-IPL MACRO
		...1		CAF2RIC	"X'10'" ..$$XH ISSUED IPW$CNC
		.... 1...		CAF2TRX	"X'08'" ..TRANSMITTER EXIT ENABLED
		.... .1..		CAF2PES	"X'04'" ..POWER IN PRIVATE ADDRESS ..SPACE AND IN ESA/370 MODE
		.... ..1.		CAF2ESA	"X'02'" ..POWER IN ESA/370 MODE
		1		CAF2MDS	"X'01'" ..NEW DEVICE STRUCTURE
(272)	626	BITSTRING	1	CAFLG3	FLAG BYTE 3
		1...		CAF3LGIG	"X'80'" ..IGNORE THE 'LOG=NO' OPTION
		.1..		CAF3QPRT	"X'40'" ..QUEUE FILE IN PART. GETVIS
		..1.		CAF3OUT0	"X'20'" ..NO ZERO PAGE LST ENTRY
		...1		CAF3Q30D	"X'10'" ..ISSUE 1Q30D AT ABN. TERMN.
		.... 1...		CAF3PHON	"X'08'" ..IDENTIFY 'PHO ESTABLISHED'
		.... .1..		CAF3NRSX	"X'04'" ..NO REAL STOR. FOR CI2
		.... ..1.		CAF3Q53I	"X'02'" ..ISSUE 1Q53I WHEN SEG'N
		1		CAF3Q41I	"X'01'" ..SUPPRESS 1Q41I
(273)	627	BITSTRING	1	CAFLG4	FLAG BYTE 4
		1...		CAF4DFCB	"X'80'" ..USE DEFAULT FCB
		.1..		CAF4IGN	"X'40'" .. SET Q-RECS WITH IGN. RECS TO DISP=Y
		..1.		CAF4SECK	"X'20'" .. ENABLE SEGMENTATION BY SPOOLING NOOP
		...1		CAF4CCW1	"X'10'" ..DO NOT ISSUE CCW X'01'
		.... 1...		CAF4WKNP	"X'08'" ..RUN NON-PARALLEL WORKUNIT
		.... .1..		CAF4QVIO	"X'04'" ..ENFORCE QFILE IN VIO
		.... ..1.		CAF4PIK	"X'02'" ..PREVENT QUEUE MANIPULATION COMMAND DURING PICKUP OF ACT ENTRY
		1		CAF4Q31	"X'01'" ..ALLOW QFILE BEYOND 16MB
(274)	627	BITSTRING	1	CAFLG5	FLAG BYTE 5
		1...		CAF5X80	"X'80'" ..USED BY HIGHER RELEASE
		.1..		CAF5OES	"X'40'" ..ALLOW OUTEXIT FOR SAS GET
		..1.		CAF5ANF	"X'20'" ..FLUSH AUTONAME JOBS
		...1		CAF5ANH	"X'10'" ..HOLD AUTONAME JOBS
		.... 1...		CAF5OCL	"X'08'" ..OUTDYNCL=DYNCL
		.... .1..		CAF5DST	"X'04'" ..RSCSROOM=DIST
		.... ..1.		CAF5IFL	"X'02'" ..INTFLUSH=OPER
		1		CAF5FTTR	"X'01'" ..FULL TASK TRACE
(275)	629	CHARACTER	2	CADFCBSF	PRT1 FCB SUFFIX,SET BY $I2, DEFAULT: 2 BLANKS
(277)	631	BITSTRING	1	CAFLG6	FLAG BYTE 6
		1...		CAF6R3F	"X'80'" ..1R33D=FLUSH
		.1..		CAF6R31	"X'40'" ..1R33D=IGNORE
		..1.		CAF6IS31	"X'20'" ..INCLUDE SVA-31 IN DUMP
		...1		CAF6PDMP	"X'10'" ..NO SETPRT SEGMENT PDUMP
		.... 1...		CAF6MAIP	"X'08'" ..PAUSING FOR MAXCL=1 PART.
(278)	632	BITSTRING	1	CAFLG7	FLAG BYTE 7
		1...		CAF7PTB	"X'80'" ..TRACE BSC NODES
		.1..		CAF7PTC	"X'40'" ..TRACE CTC NODES
		..1.		CAF7PTT	"X'20'" ..TRACE TCP NODES
		...1		CAF7PTS	"X'10'" ..TRACE SLL NODES
(279)	633	BITSTRING	3	(3)	RESERVED

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
MODULE LOAD ADDRESSES TWO TABLES OF MODULE ADDRESSES ARE MAINTAINED BY VSE/POWER. THE FIRST TABLE ADDRESSES THOSE MODULES WHICH ARE ALWAYS LOADED BY THE SYSTEM. THE SECOND ADDRESSES THOSE MODULES WHICH FORM OPTIONAL FUNCTIONS OF THE SYSTEM. THE FIRST TABLE ACTS BOTH AS A 'LOAD LIST' FOR THE INITIATOR TASK AND AS AN 'ADDRESS LIST' FOR USE IN MODULE LINKAGE. THE TABLE IS INITIALISED SO THAT EACH FULL WORD ENTRY CONTAINS THE 2- OR 3-CHARACTER SYLLABLE WHICH IDENTIFIES THE PHASE TO WHICH THE ENTRY RELATES. IT SHOULD BE NOTED THAT THE ORDER IN WHICH THE ENTRIES APPEAR IN THE LIST DETERMINES THE ORDER IN WHICH THE CORRESPONDING PHASES ARE LOADED IN THE VSE/POWER PAGEABLE AREA. AS THE INITIATOR PHASE 1 (IPW$$I1) LOADS EACH PHASE INTO STORAGE, IT RE-INITIALISES THE TABLE ENTRY TO CONTAIN THE VIRTUAL ADDRESS OF THE FIRST BYTE OF THE PHASE. IF RUNNING ON ESA/370, THIS TABLE IS UPDATED WITH THE NAMES OF 'ESA-MODULES' BY IPW$$I1. ALL PHASES ARE LOADED ON X'100' BOUNDARY. THE TABLE CAN THEN BE USED TO EFFECT LINKAGE FROM ONE VSE/POWER PHASE TO ANOTHER.					
(27C)	636	SIGNED	4	CAFM (0)	ALIGNMENT
(27C)	636	CHARACTER	4	CACP	COMMAND PROCESSOR
READER TASK PHASES					
(280)	640	CHARACTER	4	CAPD	PUT DATA RECORD
(284)	644	CHARACTER	4	CALR	LOGICAL READER
(288)	648	CHARACTER	4	CAPR	PHYSICAL READER
(28C)	652	CHARACTER	4	CASN	SCAN AND CHECK PARAMETER
WRITER TASK PHASES					
(290)	656	CHARACTER	4	CAPP	PHYSICAL PUNCH
(294)	660	CHARACTER	4	CAPL	PHYSICAL LIST
(298)	664	CHARACTER	4	CAGD	GET DATA RECORD
(29C)	668	CHARACTER	4	CALW	LOGICAL WRITER
EXECUTION PROCESSOR PHASES					
(2A0)	672	CHARACTER	4	CAXJ	JECL ANALYSIS
(2A4)	676	CHARACTER	4	CAXR	EXECUTION READER
(2A8)	680	CHARACTER	4	CAXW	EXECUTION WRITER
QUEUE MANAGEMENT PHASES					
(2AC)	684	CHARACTER	4	CADQ	DELETE FROM QUEUE CHAIN
(2B0)	688	CHARACTER	4	CAAQ	ADD TO QUEUE
(2B4)	692	CHARACTER	4	CANQ	GET NEXT FROM QUEUE
(2B8)	696	CHARACTER	4	CARQ	RESERVE QUEUE
(2BC)	700	CHARACTER	4	CAFQ	FREE QUEUE

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(2C0)	704	CHARACTER	4	CASQ	SERVICE RTN'S FOR QUEUE FILE
(2C4)	708	CHARACTER	4	CA1Q	GENERAL SERVICE ROUTINES
		MISCELLANEOUS PHASES			
(2C8)	712	CHARACTER	4	CALU	LUB/PUB UPDATE FUNCTION
(2CC)	716	CHARACTER	4	CAAS	ASYNCHRONOUS SERVICE RTN
(2D0)	720	CHARACTER	4	CATR	TASK TERMINATOR
(2D4)	724	CHARACTER	4	CAOT	OPEN TAPE ROUTINE
(2D8)	728	CHARACTER	4	CAOF	OFFLOAD MODULE
(2DC)	732	CHARACTER	4	CAER	3540 PHYSICAL READER
(2E0)	736	CHARACTER	4	CAOE	3540 OPEN ROUTINE
(2E4)	740	CHARACTER	4	CASY	SYSIN TAPE SUPPORT
(2E8)	744	CHARACTER	4	CAPS	PRINT STATUS REPORT
(2EC)	748	CHARACTER	4	CAPS1	PRINT STATUS SERVICE
(2F0)	752	CHARACTER	4	CAIC	INVOKE CP FUNCTION
(2F4)	756	CHARACTER	4	CARY	QUEUE FILE RECOVERY
(2F8)	760	CHARACTER	4	CAAT	ABNORMAL TERMINATION PROGRAM
(2FC)	764	CHARACTER	4	CALO	LOGICAL OUTPUT ROUTINE
(300)	768	CHARACTER	4	CADT	DEFINE TABLES & CNTL RECORDS
(304)	772	CHARACTER	4	CAPC	PARAMETER CHECKING RTN
(308)	776	CHARACTER	4	CADS	DATA MANAGEMENT SERVICES
(30C)	780	CHARACTER	4	CA$OP	OUTPUT PARAMETER ROUTINE
(310)	784	CHARACTER	4	CADP	DYNAMIC PART. SCHEDULER
(314)	788	CHARACTER	4	CAID	IDUMP IN FLIGHT ROUTINE
		MESSAGE PROCESSING PHASES			
(318)	792	CHARACTER	4	CAMS	MESSAGE HANDLER
(31C)	796	CHARACTER	4	CAMX	MESSSAGE DISTRIBUTOR
(320)	800	CHARACTER	4	CA$M	MESSAGE DEFINITION MODULE
		CROSS PARTITION COMMUNICATION PROCESSING PHASES			
(324)	804	CHARACTER	4	CAXM	X-PARTITION MASTER ROUTINE
(328)	808	CHARACTER	4	CAXT	X-PARTITION USER MAIN RTN
(32C)	812	CHARACTER	4	CAXTG	X-PARTITION GET FUNCTION RTN
(330)	816	CHARACTER	4	CAXTC	X-PARTITION CTL FUNCTION RTN

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(334)	820	CHARACTER	4	CAXTP	X-PARTITION PUT FUNCTION RTN
(338)	824	CHARACTER	4	CAXTS	X-PARTITION SUBROUTINES
(33C)	828	CHARACTER	4	CANY	NOTIFY SUPPORT
(340)	832	CHARACTER	4	CAXTM	X-PARTITION GCM FUNCTION DUMMY LOAD OF XTS
MISCELLANEOUS PHASES FOR UNATTENDED NODE					
(344)	836	CHARACTER	4	CAXH	X-PARTITION HEARTBEAT RTN
(348)	840	CHARACTER	4	CATQM	SUPP. WAIT FOR RUN SUBQUEUE
(34C)	844	CHARACTER	4	CATVM	SUPPORT TIME INTERVAL
		..11 .1.1		CANM	"(*-CAFM)/4" NUMBER OF MODULES
SPOOL MANAGEMENT (OPTIONAL)					
(350)	848	CHARACTER	4	CASF	SPOOL MANAGER
ACCOUNTING SUPPORT (OPTIONAL) FOR THE ACCOUNTING OPTION, UPDATED BY IPW$$I1 EITHER FOR C-K-D OR FBA.					
(354)	852	CHARACTER	4	CAPA	PUT ACCOUNT RTN (PF - FBA)
(358)	856	CHARACTER	4	CAGA	GET ACCOUNT RTN (GF - FBA)
(35C)	860	CHARACTER	4	CASA	PACCOUNT ROUTINE (SF - FBA)
(360)	864	CHARACTER	4	CABA	BUILD ACCOUNT RECORD RTN
		.... .1..		CAAM	"(*-CAPA)/4" NUMBER OF ACCOUNT MODULES
SOURCE STATEMENT LIBRARY INCLUSION (OPTIONAL)					
(364)	868	CHARACTER	4	CASL	GET SLB STATEMENT
REMOTE JOB ENTRY - RJE,BSC (OPTIONAL)					
(368)	872	CHARACTER	4	CATM	BSC LINE MANAGER ADDRESS
(36C)	876	CHARACTER	4	CABR	BSC RJE READER
(370)	880	CHARACTER	4	CABW	BSC RJE WRITER
		11		CANB	"(*-CATM)/4" NUMBER OF RJE,BSC MODULES
REMOTE JOB ENTRY - RJE,SNA (OPTIONAL)					
(374)	884	CHARACTER	4	CAS0	SNA MANAGER - SN
(378)	888	CHARACTER	4	CAS3	MESSAGE PROCESSOR - MP
(37C)	892	CHARACTER	4	CAS5	INBOUND PROCESSOR - IB
(380)	896	CHARACTER	4	CAS6	OUTBOUND PROCESSOR - OB
(384)	900	CHARACTER	4	CAS7	VTAM EXIT MODULE - VE
(388)	904	CHARACTER	4	CAS8	LOGON PROCESSOR(1) - LH
(38C)	908	CHARACTER	4	CAS2	LOGOFF PROCESSOR - LF
(390)	912	CHARACTER	4	CAS9	LOGON PROCESSOR(2) - LN

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(394)	916	CHARACTER	4	CAS10	OUTBOUND COMPACTION - OC
		.... 1..1		CANS	"(*-CAS0)/4" NUMBER OF SNA MODULES
		SHARED SPOOLING FEATURE (OPTIONAL)			
(398)	920	CHARACTER	4	CATT	TIMER TASK MODULE
		.1.. 1...		CANU	"(*-CAFM)/4" NUMBER OF POWER MODULES WITHOUT RDR EXIT
		READER EXIT (OPTIONAL)			
(39C)	924	CHARACTER	4	CARE	RDREXIT ADDRESS
		OUTPUT EXIT (OPTIONAL)			
(3A0)	928	CHARACTER	4	CAOEX	OUTPUT EXIT ADDRESS
		TRACE FACILITY (OPTIONAL)			
(3A4)	932	ADDRESS	4	CATC	ADDR OF TRACE FACILITY MOD
		TRACING FACILITY			
(3A8)	936	SIGNED	4	CATCS (0)	TRACE CONTROL/STATE
(3A8)	936	BITSTRING	2	CATCCT (0)	TRACE CONTROL (VERB)
(3A8)	936	BITSTRING	1	CATCCT1	TRACE CONTROL BYTES 1
		1...		CATCCSIM	"X'80'" ..STOP SHUTDOWN: IMMED +
		.1..		CATCCIMC	"X'40'" .. " DUE TO PSTOP CMND * @D52TDSW
		...1		CATCCIMI	"X'10'" .. " DUE TO INTERN ERR * @D52TDSW
		.... 1...		CATCCSPE	"X'08'" ..STOP PEND *
		.... .1..		CATCCSOJ	"X'04'" ..STOP EOJ *
		.... ..1.		CATCCSPA	"X'02'" ..STOP PAUSE: PSTOP CMD *
		1		CATCCSON	"X'01'" ..STOP PAUSE: ONCE BUF FUL+
		..1.		CATCCIMD	"X'20'" ..STOP PAUSE: IDUMP ERR * * = ALREADY PROCESSED INDICATOR + = INDICATED BEFORE PROCESSING
(3A9)	937	BITSTRING	1	CATCCT2	TRACE CONTROL BYTE 2
		1...		CATCC2QB	"X'80'" ..QUIESCE BUFFERS
		.1..		CATCCION	"X'40'" ..STOP ONCE BUF FULL *
(3AA)	938	BITSTRING	1		UNUSED
(3AB)	939	BITSTRING	1	CATCST	TRACE STATE INDICATOR (TESTED BY 'CLI' INST)
		1... 1111		CATCSSTP	"X'8F'" ..TRACE STOPPING (SEE ABOVE)
		.... 111.		CATCSRUN	"X'0E'" ..TRACE RUNNING ENABLED
		.... 11.1		CATCSRUD	"X'0D'" ..TRACE RUNNING DISABLED
		1... 1.11		CATCSQBU	"X'8B'" ..TRACE QUIESCING BUFFERS
		1... 1..1		CATCSQID	"X'89'" ..TRACE QUIESCING IDUMP
		.... .111		CATCSINB	"X'07'" ..TRACE RESETING BUFFERS
		.... .1.1		CATCSINI	"X'05'" ..TRACE INITIAL(IZING)/PAUSE
		1... ..11		CATCSPRS	"X'83'" ..TRACE PREPARED + STOPPING
		11		CATCSPRE	"X'03'" ..TRACE PREPARED
		.... ..1.		CATCSSTR	"X'02'" ..TRACE STARTED
		1		CATCSDOR	"X'01'" ..TRACE DORMANT (STOPPED)
		....		CATCSDWN	"X'00'" ..TRACE DOWN (NEVER STARTED)
(3AC)	940	SIGNED	4	(0)	TRACE CONFIGURATION/MODE
(3AC)	940	BITSTRING	1	CATCFIG (0)	
(3AC)	940	BITSTRING	1	CATCFIG1	
		1...		CATCFBWR	"X'80'" ..TRACE BUFFERING WRAPPED
		.1..		CATCFBID	"X'40'" ..TRACE BUFFERING IDUMP
		..1.		CATCFBON	"X'20'" ..TRACE BUFFERING ONCE
		...1		CATCFBUR	"X'10'" ..TRACE BUF REAL (VIRT=OFF)
		.... 1...		CATCFBPR	"X'08'" ..TRACE BUFFERING PSEUDO REAL (NO PG FLT HANDL'G>>FUTURE<<
		.... ..1.		CATCFLOD	"X'02'" ..TRACE PHASES LOADED
		1		CATCFTVD	"X'01'" ..TRACE TABLE VALID
(3AD)	941	BITSTRING	1	CATCFIG2	
		1...		CATCF2WM	"X'80'" ..ISSUE WRAP BUF FULL MSG

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
		.1..		CATCF2ST	"X'40'" ..MAINTAIN TRACE STATISTICS
		..1.		CATCF2XC	"X'20'" ..ACTIVATE XPCC RCV/RP TRACE
(3AE)	942	BITSTRING	2		UNUSED
(3B0)	944	SIGNED	4	(0)	MISCELLANEOUS
(3B0)	944	ADDRESS	4	CATCCB	ADDR OF CONTROL BLOCK(TRCB)
(3B4)	948	ADDRESS	4		USUSED
(3B8)	952	ADDRESS	4	CATCSELT	ADDR OF SELECTION TABLE
(3BC)	956	ADDRESS	4	CATCBUF1	BUFFER 1 ADDRESS
(3C0)	960	ADDRESS	4	CATCBUN1	BUFFER 1 APPROX NEXT ENTRY
(3C4)	964	ADDRESS	4	CATCBUF2	BUFFER 2 ADDRESS
(3C8)	968	ADDRESS	4	CATCBUN2	BUFFER 2 APPROX NEXT ENTRY
(3CC)	972	ADDRESS	4	CATCBUF3	(BUFFER 3 ADDRESS)
(3D0)	976	ADDRESS	4	CATCBUN3	(BUFFER 3 NEXT ENTRY)
(3D4)	980	SIGNED	2	CATCSELZ	SIZE OF SELECTION TBL AREA
(3D6)	982	SIGNED	2	CATCMODZ	SIZE OF MODULE PFIXED AREA
(3D8)	984	ADDRESS	4	CATCCMLK	TRACE COMMAND LOCKWORD: X'80000000' ..RESOURCE 'LIVE' BIT X'00FFFFFF' ..LAST USER'S ADDRESS (GATES ALL TRACE COMMANDS EXCEPT "PSTOP IMMEDIATE")
(3DC)	988	ADDRESS	4	(4)	UNUSED
(3DC)	988	BITSTRING		CATCTBLN	"X'1400'" MIN SIZE OF TRACE SELEC TBL
(3DC)	988	BITSTRING		CATCLOAD	"X'4800'+CATCTBLN" SIZE OF TRACE IPW$$TC + SELECTION TBL IPWSTBL FOR INIT CHECK
		...1 .11.		CATCENTR	"X'16'" IPW$$TC ENTRY DISP: TRACING
		.... 1...		CATCRTND	"X'08'" IPW$$TC TRACING RETURN TO CALLER DISP I.E. RETURN R4 + DISP
		..11		CATCVER	"X'30'" IPW$$TC VER/MOD LOCATION
		...1 ...1		CATCSLVM	"X'11'" SELECTION TABLE VER/MOD LOC
		LENGTH OF NUCLEUS			
(3EC)	1004	SIGNED	4	CANULN	LENGTH OF NUCLEUS
(3F0)	1008	ADDRESS	4	CAEXTAB	ADDRESS OF EXIT DATA TABLE
(3F4)	1012	ADDRESS	4	CAFCTAB	ADDRESS OF FCB TABLE
		MISCELLANEOUS			
(3F8)	1016	ADDRESS	4	CAPFCF	ADDR(CURRENT PAGE FAULT REQ)
(3FC)	1020	ADDRESS	4	CANUCS	ADDR(NUCS 15C DY.....)

<table>
<tr><th>Offset Hex</th><th>Offset Dec</th><th>Type</th><th>Len</th><th>Name (Dim)</th><th>Description</th></tr>
<tr><td colspan="6">SERVICE ROUTINE BRANCH TABLE
THE FOLLOWING TABLE IS USED TO ESTABLISH THE ENTRY POINTS
OF THE VSE/POWER TASK MANAGEMENT ROUTINE CONTAINED IN THE
VSE/POWER NUCLEUS PHASE. SINCE ALL SERVICES HAVE DIRECT
ADDRESSABILITY BY MEANS OF REG. 10 THE TABLE CONSISTS OF
A SET OF BRANCH INSTRUCTIONS.
EACH VSE/POWER TASK MANAGEMENT SERVICE MACRO INSTRUCTION
GENERATES A BRANCH AND LINK TO THE APPROPRIATE ENTRY
WITHIN THIS BRANCH TABLE. TASK MANAGEMENT SERVICES
HAVE NO ADDRESSABILITY INTO THE R9 AND R8 BASE
REGISTER AREA OF IPW$$NU!
THE FOLLOWING BRANCH TABLE ALLOWS NUCLEUS SERVICE ROUTINES
TO RESIDE OUTSIDE THE FIRST 4K OF THE NUCLEUS AS ADDRESSED
VIA REGISTER 10. THESE ROUTINES (IF CALLED FROM OUTSIDE
THE NUCLEUS) WILL SAVE REGISTER 9 IN THE FIELD 'TC09'
AND REGISTER 8 IN FIELD TC08 OF THE CALLER'S TCB AND
WILL SET UP REGISTER 9,8 AS THE SECOND AND THIRD BASE
REGISTER FOR THE VSE/POWER NUCLEUS. UPON EXIT FROM
NUCLEUS, R9 AND R8 ARE AGAIN RESTORED FOR TASK USE.
WHEN CALLED FROM WITHIN THE NUCLEUS, REGISTER 9 AND 8
ARE NEITHER SAVED NOR RELOADED AT EXIT, BECAUSE THEY
CONTAIN ALREADY THE CORRECT BASE ADDRESSES. THIS
DETERMINATION IS MADE BASED ON THE LINK REGISTER
(R0, OR R2). IF THE LINK IS VIA BRANCH AND LINK
THEN THE HIGH-ORDER BYTE OF THE LINK REGISTER IS NOT ZERO.
THE VSE/POWER MACROS EXPAND TO BRANCH AND LINK FOR
CALLS FROM OUTSIDE NUCLEUS. THE GLOBAL '&NUSA' IS SET
ON NUCLEUS ASSEMBLY AND THE VSE/POWER NUCLEUS CALLS
THEN EXPAND TO SET LINK REGISTER VIA 'LOAD ADDRESS'
INSTRUCTION. THIS INSURES THAT THE HIGH-ORDER
BYTE OF THE LINK REGISTER IS ZERO, THEREBY GIVING US
A SWITCH BY WHICH TO DETERMINE FROM WHERE WE WERE CALLED.
THE ADDRESS CONSTANTS ARE RELOCATED BY THE VSE LOADER.</td></tr>
<tr><td>(410)</td><td>1040</td><td>SIGNED</td><td>4</td><td>(0)</td><td>ALIGNMENT</td></tr>
<tr><td>(414)</td><td>1044</td><td>ADDRESS</td><td>4</td><td></td><td>RESERVE RESOURCE</td></tr>
<tr><td>(41C)</td><td>1052</td><td>ADDRESS</td><td>4</td><td></td><td>RELEASE RESOURCE</td></tr>
<tr><td>(424)</td><td>1060</td><td>ADDRESS</td><td>4</td><td></td><td>RESERVE WORK SPACE</td></tr>
<tr><td>(42C)</td><td>1068</td><td>ADDRESS</td><td>4</td><td></td><td>RELEASE WORK SPACE</td></tr>
<tr><td>(434)</td><td>1076</td><td>ADDRESS</td><td>4</td><td></td><td>MESSAGE SERVICE LOCAL</td></tr>
<tr><td>(43C)</td><td>1084</td><td>ADDRESS</td><td>4</td><td></td><td>MESSAGE SERVICE REMOTE</td></tr>
<tr><td>(444)</td><td>1092</td><td>ADDRESS</td><td>4</td><td></td><td>TIMER INTERVAL SERVICE RTN</td></tr>
<tr><td>(44C)</td><td>1100</td><td>ADDRESS</td><td>4</td><td></td><td>NODAL MESSAGE SERVICE RTN</td></tr>
<tr><td>(454)</td><td>1108</td><td>ADDRESS</td><td>4</td><td></td><td>DISK SERVICE</td></tr>
<tr><td>(45C)</td><td>1116</td><td>ADDRESS</td><td>4</td><td></td><td>NOTIFY MESSAGE SERVICE</td></tr>
<tr><td>(464)</td><td>1124</td><td>ADDRESS</td><td>4</td><td></td><td>TAPE SERVICE</td></tr>
<tr><td>(46C)</td><td>1132</td><td>ADDRESS</td><td>4</td><td></td><td>TIMER SERVICE</td></tr>
<tr><td>(474)</td><td>1140</td><td>ADDRESS</td><td>4</td><td></td><td>VALIDATE DATA AREA ADDRESS</td></tr>
<tr><td>(47C)</td><td>1148</td><td>ADDRESS</td><td>4</td><td></td><td>RETRIEVE MESSAGE TEXT</td></tr>
<tr><td>(484)</td><td>1156</td><td>ADDRESS</td><td>4</td><td></td><td>RESERVE VIRTUAL STORAGE</td></tr>
<tr><td>(48C)</td><td>1164</td><td>ADDRESS</td><td>4</td><td></td><td>RELEASE VIRTUAL STORAGE</td></tr>
<tr><td>(494)</td><td>1172</td><td>ADDRESS</td><td>4</td><td></td><td>UNCHAIN ELEMENT</td></tr>
<tr><td>(49C)</td><td>1180</td><td>ADDRESS</td><td>4</td><td></td><td>SET REMOTE MASK ROUTINE</td></tr>
<tr><td>(4A4)</td><td>1188</td><td>ADDRESS</td><td>4</td><td></td><td>GET TRACE ENTRY ROUTINE</td></tr>
<tr><td>(4AC)</td><td>1196</td><td>ADDRESS</td><td>4</td><td></td><td>QUEUE FILE SERVER</td></tr>
<tr><td>(4BC)</td><td>1212</td><td>ADDRESS</td><td>4</td><td>(3)</td><td>RESERVED FOR TRACE FACILITY</td></tr>
<tr><td>(4CC)</td><td>1228</td><td>ADDRESS</td><td>4</td><td></td><td>NP/PU MODE SWITCH SERVICE</td></tr>
<tr><td>(4D4)</td><td>1236</td><td>ADDRESS</td><td>4</td><td></td><td>DOM MESSAGE SERVICE</td></tr>
<tr><td colspan="6">STATISTICAL INFORMATION
THE FOLLOWING FIELDS ARE USED TO MAINTAIN THE STATISTICAL
INFORMATION FOR INCORPORATION IN THE STATUS REPORT PRODUCED
AT SHUT-DOWN TIME.</td></tr>
<tr><td>(4D8)</td><td>1240</td><td>SIGNED</td><td>2</td><td>NRRE</td><td>HIGHEST BSC REMID</td></tr>
<tr><td>(4DA)</td><td>1242</td><td>SIGNED</td><td>2</td><td>NRLI</td><td>NUMBER OF BSC LINES</td></tr>
</table>

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(4DC)	1244	SIGNED	4	NRTR	TOTAL NR OF TRACKS DATA FILE
(4DC)	1244	SIGNED	4	NBLK	TOTAL NR OF BLOCKS
(4E0)	1248	SIGNED	4	NRTW	TIMES WAITING FOR REAL STORAGE
(4E4)	1252	SIGNED	4	NRTV	TIMES WAITING FOR VIRT.STORG
(4E8)	1256	SIGNED	4	NRPG	TOTAL NR OF PAGES ALLOCATED
(4EC)	1260	SIGNED	4	NRPC	CURRENT NR OF PAGES ALLOCATED
(4F0)	1264	SIGNED	4	NRPM	MAXIMUM NR OF PAGES ALLOCATED
(4F4)	1268	SIGNED	4	NRTC	CURRENT NR OF TASKS
(4F8)	1272	SIGNED	4	NRTH	MAXIMUM NR OF TASKS
(4FC)	1276	SIGNED	4	NRSET	PRESENT SESSION START TIME
(500)	1280	CHARACTER	8	NRSED	PRESENT SESSION START DATE
(508)	1288	SIGNED	2	NRSVA	SYSTEM GETVIS STORAGE
(50A)	1290	CHARACTER	2	NRCEN	CENTURY OF PRESENT SESSION
(50C)	1292	ADDRESS	2	NRMSAS	MAX. NO.SAS TASKS ALLOWED,OVERWRITE BY 'PVARY MAXSAS'
(50E)	1294	SIGNED	2	NRCSAS	CURR.NO.SAS TASKS ACTIVE
(510)	1296	ADDRESS	4	NRSASDOM	DOM-ID FOR MSG 1Q3JA

MISCELLANEOUS DEFINITIONS
THE FOLLOWING 2 WORDS ARE USED TO IDENTITY THE SIZE OF
WORKSPACE REQUIRED TO ACCOMODATE PHYSICAL AND LOGICAL
DATA AREAS.

(514)	1300	SIGNED	4	CABLBF	PHYSICAL DATA BUFFER SIZE
(518)	1304	SIGNED	4	CABLDB	LOGICAL DATA BUFFER SIZE
(51C)	1308	ADDRESS	4	CAOPDE	ADDRESS OF 1ST OPDE IF ANY
(520)	1312	ADDRESS	4	CATTRA	ADDRESS OF TASK TRACE AREA
(524)	1316	CHARACTER	8	CAMPWD	ENC/DEC SYSDATE

RJE,BSC CONTROL BLOCK ADDRESSES

(52C)	1324	ADDRESS	4	CALC	FIRST LINE CONTROL BLOCK ADDRESS
(530)	1328	ADDRESS	4	CART	BSC REMOTE TABLE ADDRESS
(534)	1332	ADDRESS	4	CALT	BSC LINE TABLE ADDRESS

DEBUG SWITCH BYTE
SET NORUN SWITCH BYTE
VSE/POWER PHASE IPW$$I1 COPIES THE UPSI BYTE INTO THE CAT
TO MAKE IT EASY ADDRESSABLE FOR EVERY VSE/POWER ROUTINE.
ACTIVATION OF SET NORUN=YES WITH VSE JCL CARD
// UPSI 1 IN VSE/POWER STARTUP DECK OR BY AR
COMMAND PAUSE.

(538)	1336	BITSTRING	1	CAUP	COPY OF UPSI BYTE (P.COMRG)
		1...		CAU1	"X'80'" SET NORUN=YES ACTIVE

IF SWITCH IS ON SET NORUN=YES WILL BE ACTIVATED
IN IPW$$I1

		.1..		CAU2	"X'40'" LOG PNET I/O ON CONSOLE

IF SWITCH IS ON, ALL I/OS FOR A PNET BSC LINE WHICH HAS
BEEN STARTED WITH THE TRACE OPTION, ARE DISPLAYED ON
THE SYSTEM CONSOLE.

		..1.		CAU3	"X'20'" DUMP TRACE AREA IF FILLED UP

IF SWITCH IS ON, THE TRACE AREA IS DUMPED TO THE SYSTEM
DUMP FILE.

		...1		CAU4	"X'10'" TRACE SAS & DST TASK EVENTS

IF SWITCH IS ON, ALL CROSS PARTITION EVENTS, SUCH
AS A RECEIVE OR REPLY ARE TRACED

		.... 1...		CAU5	"X'08'" LOAD INTERNAL TRACE FACILITY

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
IF SWITCH IS ON, THEN THE TRACE MODULE IPW$$TC AND THE TRACE DEFAULT SELECTION TABLE IPW$TBL IS LOADED TO THE PFIXED AREA ALLOWING THE INTERNAL TRACE FACILITY TO RUN.					
(539)	1337	BITSTRING	1	CAMD	HARDWARE MACHINE MODE C' ' .. /370 MODE OR ESA MODE
		11.. .1.1		EMOD	"X'C5'" .. E MODE
(53A)	1338	CHARACTER	6	CAVOL	QUEUE FILE DISK VOLID
(540)	1344	ADDRESS	2	CABLK	MAXIMUM DBLK VALUE
(542)	1346	BITSTRING	2	CACAS	CURRENT ADDRESS SPACE ID
(544)	1348	ADDRESS	4	CASYMP	ADDR FOR SYMPTOM RECORD
(548)	1352	CHARACTER	4	CAUNALST	SYSLST ASSIGNMENT BEFORE POWER UNASSIGNS IT
(54C)	1356	ADDRESS	4	CAPRPBG	1ST BYTE OF PRIV ADDR SPACE
(550)	1360	CHARACTER	1	CAHOLDC	SET 'HOLDCL=CLASS' VALUE
(551)	1361	ADDRESS	3		RESERVED FOR FUTURE USE
(554)	1364	ADDRESS	4	CASTXOC	ADDR OF STXIT OC ROUTINE
(554)	1364			CATLN1	"*" END OF CONTROL PART OF CAT
EQUATE STATEMENTS THE FOLLOWING EQUATE STATEMENTS PROVIDE NECESSARY RESOLUTION FOR UNDEFINED SYMBOLS WITHIN THE PADS DUMMY SECTION					
		....		AI00	"0" ATTENTION INTERFACE
		....		PF00	"0" PAGE FAULT APPENDAGE
		....		HR00	"0" HOT READER ROUTINE
		....		CE00	"0" RJE CE ROUTINE
		....		SU00	"0" SVC 0 INTERFACE
		....		SU00ES	"0" SVC 0 INTERFACE IN ESA MODE
		....		SU90	"0" SVC 90 INTERFACE
		....		EOJ00	"0" JCL END OF JOB EXIT ROUTINE
		....		SEG00	"0" IPWSEGM INTERFACE ROUTINE
		....		FTTR00	"0" FULL TASK TRACE ROUTINE
		....		TI00	"0" INTERVAL TIMER ROUTINE
		....		STXOC000	"0" STXIT OC ROUTINE
		....		CMNDTAB	"0" ADDRESS OF COMMAND TABLE
		....		CMND@SEL	"0" ADDRESS OF COMMAND TABLE
		....		SCBD	"0" STORAGE CONTROL BLOCK
		....		MMBD	"0" MESSAGE CONTROL BLOCK
		....		TM10	"0" TASK MANAGEMENT
		....		TM30	"0" TASK MANAGEMENT
		....		TM40	"0" TASK MANAGEMENT
		....		TM50	"0" TASK MANAGEMENT
		....		TM55	"0" TASK MANAGEMENT
		....		TM60	"0" TASK MANAGEMENT
		....		TM80	"0" TASK MANAGEMENT
		....		TM90	"0" TASK MANAGEMENT
		....		TM20	"0" TASK MANAGEMENT
		....		TMB0	"0" TASK MANAGEMENT
		....		TMT0	"0" TASK MANAGEMENT
		....		TMTC	"0" WAIT CONTROL BLOCK
		....		ITTC	"0" INITIATOR TCB
		....		TM02	"0" TASK MANAGEMENT
		....		TA01	"0" ATTACH TASK
		....		TD01	"0" DETACH TASK
		....		TM01	"0" TASK SELECTION
		....		RM01	"0" RESERVE RESOURCE
		....		RM51	"0" RELEASE RESOURCE
		....		SM01	"0" RESERVE WORK SPACE
		....		SM51	"0" RELEASE WORK SPACE
		....		MM01	"0" MESSAGE MANAGEMENT LOCAL
		....		MM51	"0" MESSAGE MANAGEMENT REMOTE
		....		DM20	"0" DISK SERVICE
		....		TP20	"0" TAPE SERVICE

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
		....		TR01	"0" TIMER SERVICE
		....		VA01	"0" VALIDATE DATA AREA ADDRESS
		....		GM10	"0" RETRIEVE MESSAGE TEXT
		....		COM0	"0" COMMON R0 ENTRY POINT
		....		COM2	"0" COMMON R2 ENTRY POINT
		....		SR10	"0" SET REMOTE MASK ENTRY
		....		NS10	"0" NOTIFY MESSAGE SERVICE
		....		NM10	"0" NODAL MESSAGE SERVICE
		....		TS25	"0" TIMER INTERVAL SERVICE
		....		VS01	"0" RESERVE VIRTUAL STORAGE
		....		VS51	"0" RELEASE VIRTUAL STORAGE
		....		VS91	"0" UNCHAIN ELEMENT
		....		TZ10	"0" GET TRACE ENTRY
		....		QF10	"0" QUEUE FILE SERVER
		....		PFCF	"0" CURRENT PAGE FAULT REQUEST
		....		MD10	"0" DOM MESSAGE SERVICE
		....		PN10	"0" NP/PU MODE SWITCH SERVICE
		....		CATCXX	"0" TRACE FACILITY LBL FOR CATC
		....		VSCN2	"0" ADDRESS NUCS EYE CATCHER
		GENERAL CONSTANTS			
(558)	1368	SIGNED	4	CF01	
(55C)	1372	SIGNED	4	CF04	
(560)	1376	SIGNED	4	CF08	
(564)	1380	SIGNED	4	CF10	
(568)	1384	SIGNED	4	CF24	
		TRANSLATION TABLES			
	THE FOLLOWING TRANSLATION TABLE IS USED TO SCAN SEQUENCES OF BLANK CHARACTERS FOR THE FIRST NON-BLANK CHARACTER.				
(56C)	1388	BITSTRING	16	TRTB	
(57C)	1404	BITSTRING	16		
(58C)	1420	BITSTRING	16		
(59C)	1436	BITSTRING	16		
(5AC)	1452	BITSTRING	16		
(5BC)	1468	BITSTRING	16		
(5CC)	1484	BITSTRING	16		
(5DC)	1500	BITSTRING	16		
(5EC)	1516	BITSTRING	16		
(5FC)	1532	BITSTRING	16		
(60C)	1548	BITSTRING	16		
(61C)	1564	BITSTRING	16		
(62C)	1580	BITSTRING	16		
(63C)	1596	BITSTRING	16		
(64C)	1612	BITSTRING	16		
(65C)	1628	BITSTRING	16		
	THE FOREGOING TABLE IS ALSO USED AS A SOURCE OF BLANK CHARACTERS FOR VARIOUS PROGRAM PURPOSES.				
(5AD)	1453	CHAR-ACTER	128	BLNK (0)	IDENTIFY BLANK SOURCE
	THE FOLLOWING TRANSLATION TABLE IS USED TO SCAN SEQUENCES OF NON-BLANK CHARACTERS FOR THE FIRST BLANK CHARACTER				
(66C)	1644	BITSTRING	16	TRTC	
(67C)	1660	BITSTRING	16		
(68C)	1676	BITSTRING	16		
(69C)	1692	BITSTRING	16		
(6AC)	1708	BITSTRING	16		
(6BC)	1724	BITSTRING	16		
(6CC)	1740	BITSTRING	16		
(6DC)	1756	BITSTRING	16		
(6EC)	1772	BITSTRING	16		
(6FC)	1788	BITSTRING	16		
(70C)	1804	BITSTRING	16		

<table>
<tr><th>Offset Hex</th><th>Offset Dec</th><th>Type</th><th>Len</th><th>Name (Dim)</th><th>Description</th></tr>
<tr><td>(71C)</td><td>1820</td><td>BITSTRING</td><td>16</td><td></td><td></td></tr>
<tr><td>(72C)</td><td>1836</td><td>BITSTRING</td><td>16</td><td></td><td></td></tr>
<tr><td>(73C)</td><td>1852</td><td>BITSTRING</td><td>16</td><td></td><td></td></tr>
<tr><td>(74C)</td><td>1868</td><td>BITSTRING</td><td>16</td><td></td><td></td></tr>
<tr><td>(75C)</td><td>1884</td><td>BITSTRING</td><td>16</td><td></td><td></td></tr>
<tr><td colspan="6">THE FOREGOING TABLE IS ALSO USED AS A SOURCE OF ZERO
CHARACTERS FOR VARIOUS PROGRAM PURPOSES.</td></tr>
<tr><td>(6EC)</td><td>1772</td><td>CHARACTER</td><td>128</td><td>ZERO</td><td>IDENTIFY ZERO SOURCE</td></tr>
<tr><td colspan="6">REMOTE TERMINAL/WORKSTATION CONNECT TABLE
THE CONTENT OF THE FOLLOWING TABLE REPRESENTS THE REMOTE IS'S
OF ALL TERMINALS CURRENTLY LOGGED ON AT THIS SYSTEM. IT IS
USED AS A MASK FOR THE REMOTE TABLE OF THE DISK MANAGEMENT
BLOCK (DMB) TO SELECT ONLY THOSE REMOTE ID'S WHICH ARE
RELEVANT FOR THIS CPU.
EACH REMOTE ID IS REPRESENTED BY A BIT. THE BIT IS ON IF
THE TERMINAL/WORKSTATION IS LOGGED ON.</td></tr>
<tr><td>(76C)</td><td>1900</td><td>BITSTRING</td><td>1</td><td>RIDTAB (32)</td><td>REMOTE ID TABLE (0-255)</td></tr>
<tr><td>(78C)</td><td>1932</td><td>BITSTRING</td><td>1</td><td>(32)</td><td>SPACE RESERVED (256-511)</td></tr>
<tr><td>(7AC)</td><td>1964</td><td>ADDRESS</td><td>4</td><td>(12)</td><td>RESERVED</td></tr>
</table>

How to Locate: Refer to Figure 151 on page 730 in Chapter 6, “Diagnostic Aids.”

Data Set Control Block

The data set control block is created by the network receiver for each queue entry to be allocated. Its contents are:

1. Spool control information
2. Queue entry characteristics

Bytes	Label	
Hex.	of Field	Description/Function
000-00F	DSDESCR	Storage descriptor
010-013	DSNEXT	Address next DSCB entry
014-02B	DSTCDAST	Data file status field
014-017	DSTCBDW	• Relative DBLK number
018-01B	DSTCBDV	• Address logical data buffer
01C-01D		• Data area length
01E-01F		• Flag/Operation byte
020-023	DSTCBBC	• Residual block count
024-027	DSTCBPR	• Previous record address
028-02B	DSTCBAS	• Address of Spool Environment Block
02C-037	DSTCQFST	Queue file status fields
02C-02F	DSTCBQW	• Relative queue record number
030-033	DSTCBQV	• Queue space address
034-035		• Data area length
036-037		• Flag/Operation byte
038-06F	DSQREC	DSCB characteristics field
038-039		• Not used
03A	DSQRPY	• Job priority
03B	DSQRQI	• Queue record identifier
03C	DSQRDP	• Disposition
03D	DSQRCL	• Class
03E	DSQRNC	• Number of copies
03F-042	DSQRFI	• Forms ID
043-046	DSQRCP	• Compaction table name
047-054	DSQR3800	• 3800 characteristics
047-04A	DSQRFL	• Forms-overlay identifier
04B-052	DSQRCG	• 8 copy groups
053	DSQRGI	• Copy group index
054	DSQRPS	• Burst mode indicator
055-05C	DSQRTN	• Target node name
05D-064	DSQRTU	• Target user ID
065-06C	DSFCB	• FCB name
06D-06F		
070-075	DSOPTB	Output processing fields
070-073	DSOPTBAD	• Address of OPTB structure
074-075	DSOPTBLN	• Length of OPTB structure

Compaction Table Block (CMPT)

Definition Macro: IPW$DVD CMPT=YES

The compaction table block is initialized in IPW$$LD2 whenever a valid Function Management Header 3 is received. The compaction table is built using the master and nonmaster characters from the Function Management Header 3.

Bytes Hex.	Label of Field	Description/Function
000-113	CMPTDS	Compaction Table Block
000-00F	CMPTSD	Storage descriptor (CMPT)
010	CMPTMAST	Number of master characters
011-013		Reserved for future use
014-113	CMPTTAB1	Compaction Table

Disk Management Block (DMB)

Definition Macro: IPW$DQC

The disk management block area is used to control access to the VSE/POWER queue and data file. It consists of a set of areas which collectively describe the current state of the VSE/POWER queues. The disk management block is initialized at VSE/POWER startup time (IPW$$I3) and located in the fixable area.

The disk management block is divided into the following areas:

- Resource control fields
- Record control fields
- VSE/POWER communication area
- Auxiliary queue record area
- Master record area
- Master class table area.

The format of the disk management block is shown below.

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
DISK MANAGEMENT BLOCK (DMB)					
(0)	0	CHAR-ACTER	16	QCSD	SECTION DESCRIPTOR
(10)	16	BITSTRING	12		RESERVED
(1C)	28	SIGNED	4	QCLK	LOCK WORD
RECORD CONTROL FIELDS					
THE FOLLOWING FIELDS CONTAIN THE INFORMATION USED TO READ AND WRITE RECORDS TO AND FROM THE MASTER RECORD AREA, THE AUXILARY QUEUE RECORD AREA AND THE QUEUE CONTROL AREA.					
MASTER RECORD I/O REQUEST WORD					
(20)	32	ADDRESS	4	QCMW	REL NUMBER OF MASTER RECORD
(24)	36	ADDRESS	4		VIRT ADDRESS OF MASTER REC
(28)	40	ADDRESS	2		LENGTH AND OPERATION BYTES
AUXILIARY QUEUE RECORD I/O REQUEST WORD					
(2C)	44	ADDRESS	4	QCQW	REL NUMBER OF QUEUE RECORD
(30)	48	ADDRESS	4		VIRT ADDRESS OF AUX Q-REC
(34)	52	ADDRESS	2		LENGTH AND OPERATION BYTES
I/O REQUEST WORD FOR DATA FILE ACCESS					
(38)	56	ADDRESS	4	QCDW	REL NUMBER OF DATA BLOCK
(3C)	60	ADDRESS	4		VIRT ADDRESS OF SER
(40)	64	ADDRESS	2		LENGTH AND OPERATION BYTES
(44)	68	ADDRESS	4	QCADW	CURR SLOT DBLK NUMBER
(48)	72	ADDRESS	4	QCADV	VIRTUAL SLOT DBLK ADDRESS
(4C)	76	ADDRESS	2		LENGTH AND OPERATION BYTES
(50)	80	BITSTRING	1	QCAFLG	FLAG BYTE
		1...		QCAFWN	"X'80'" .. ACTIVITY CHANGE, WRITE IT
		.1..		QCAFDR	"X'40'" .. 1ST SLOT-DBLK IN STORAGE
		..1.		QCAFDW	"X'20'" .. DO NOT WRITE BACK DBLK
		...1		QCAFDS	"X'10'" .. DELETE SLOT
(51)	81	BITSTRING	3		UNUSED
(54)	84	ADDRESS	4	QCAVIO	ADDRESS OF VIORB
(54)	84	ADDRESS	4	QCAPART	Q-FILE ADDR. IN PART. GETVIS
VSE/POWER COMMUNICATION AREA					

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(58)	88	CHARACTER	8	MRDY	DATE (CURRENT SYSTEM FORMAT)
(60)	96	SIGNED	4	MRST	VSE/POWER START TIME

OPTION SWITCH FIELDS

THE FOLLOWING SWITCH BYTES PRESERVE THE OPTIONS ESTABLISHED BY THE VSE/POWER USER AT THE TIME AT WHICH HE GENERATED HIS VERSION OF THE SYSTEM.

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(64)	100	CHARACTER	1	MRSL	SOURCE LIBRARY SWITCH This byte contains a single alphabetic character representing the source sublibrary to be searched.
(65)	101	CHARACTER	1	MRJA	ACCOUNTING SWITCH This byte contains a single alphabetic character The character A indicates that VSE/POWER job accounting is required; a blank character indicates that VSE/POWER accounting is not required.
(66)	102	CHARACTER	1	MRPP	PAUSE PUNCH SWITCH
(67)	103	CHARACTER	1	MRLG	JOB LOGGING SWITCH
(68)	104	CHARACTER	1		UNUSED
(69)	105	BITSTRING	1	MROP	GENERAL OPTION BYTE 1
		1...		MRCP	"X'80'" .. CLEAR PRINT AT EOJ
		.1..		MRMF	"X'40'" .. MARK FORM FOR SEP PAGES
		..1.		MRNS	"X'20'" .. NO SEP PAGES BTWN COPIES
					X'10' ..(USED IN QUEUE RECORD)
					X'08' .. RESERVED
					X'04' .. RESERVED
		.... ..1.		MRCH	"X'02'" .. CHANNEL 12 OPTION
		1		MRFD	"X'01'" .. FEED OPTION 3540
(6A)	106	BITSTRING	2		UNUSED

STANDARD DEFAULT FIELDS

THE FOLLOWING FIELDS CONTAIN STANDARD VSE/POWER DEFAULT VALUES USED IN CREATION OF NEW RECORDS.

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(6C)	108	CHARACTER	8	MRNM	DEFAULT JOB NAME These 8 bytes contain the character string 'AUTONAME' used as a default job name.
(74)	116	CHARACTER	1	MRCL	DEFAULT CLASS ATTRIBUTE This byte contains the alphabetic character A representing the class attribute to be given by default to each RDR queue entry created within VSE/POWER.
(75)	117	CHARACTER	1	MRPY	DEFAULT PRIORITY ATTRIBUTE This byte contains numeric character 3 which defines the priority attribute to be given by default to each queue entry created by VSE/POWER.
(76)	118	CHARACTER	2	MRDYC	CENTURY OF CURRENT DATE

MASTER LINE TABLE

The next 16-byte field contains the master line table, consisting of system default values used to analyze space and skip operations during printer control carriage simulation.

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(78)	120	CHARACTER	16	MRLT (0)	LINE TABLE
(78)	120	SIGNED	2		RESERVED
(7A)	122	SIGNED	2		DEFAULT PAGE SIZE
(7C)	124	SIGNED	1		SKIP TO CHANNEL ONE
(7D)	125	SIGNED	1		SKIP TO CHANNEL TWO
(7E)	126	SIGNED	1		SKIP TO CHANNEL THREE
(7F)	127	SIGNED	1		SKIP TO CHANNEL FOUR
(80)	128	SIGNED	1		SKIP TO CHANNEL FIVE
(81)	129	SIGNED	1		SKIP TO CHANNEL SIX
(82)	130	SIGNED	1		SKIP TO CHANNEL SEVEN
(83)	131	SIGNED	1		SKIP TO CHANNEL EIGHT

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(84)	132	SIGNED	1		SKIP TO CHANNEL NINE
(85)	133	SIGNED	1		SKIP TO CHANNEL TEN
(86)	134	SIGNED	1		SKIP TO CHANNEL ELEVEN
(87)	135	SIGNED	1		SKIP TO CHANNEL TWELVE
	MASTER LIST VALUES				
		The next 16 bytes contain the master list values, which will be inserted by default in list queue records, unless overridden by a JECL LST statement. Values are set by IPW$$IP using those specified by user during VSE/POWER generation (for example: JSEP=, RBS=, STDLINE=).			
(88)	136	CHAR-ACTER	16	MRLV (0)	MASTER LIST VALUES
(88)	136	BITSTRING	1	LV#PERF	NO. OF PERFORMATION LINES
(89)	137	BITSTRING	1		RESERVED
(8A)	138	BITSTRING	1	LVFLG	FLAG BYTE
		1...		LVSKIPIN	"X'80'" ..SKIP TO CH1 INSERTION REQ.
(8B)	139	SIGNED	1	LVSP	NUMBER OF SEPARATORS
(8C)	140	SIGNED	4	LVBS	RECORDS BEFORE SEGMENTATION
(90)	144	SIGNED	4	LVBM	RECORDS BEFORE MESSAGE
(94)	148	SIGNED	4	LVBN	RECORDS BEFORE NEXT MESSAGE
	MASTER PUNCH VALUES				
		The next 16 bytes contain the master punch values, which will be inserted by default in punch queue records, unless overridden by a JECL PUN statement. Values set by IPW$$IP using those specified by user during VSE/POWER generation (JSEP=, RBS=, STDCARD=).			
(98)	152	CHAR-ACTER	16	MRPV (0)	MASTER PUNCH VALUES
(98)	152	SIGNED	3		RESERVED
(9B)	155	SIGNED	1	PVSP	NUMBER OF SEPARATORS
(9C)	156	SIGNED	4	PVBS	RECORDS BEFORE SEGMENTATION
(A0)	160	SIGNED	4	PVBM	RECORDS BEFORE MESSAGE
(A4)	164	SIGNED	4	PVBN	RECORDS BEFORE NEXT MESSAGE
	TIMER TASK VALUES (SHARED SPOOLING)				
(A8)	168	ADDRESS	4	MREB	SHARED SPOOLING SUBTASK ECB
(AC)	172	CHAR-ACTER	8	MRTI (0)	TIMER INTERVAL VALUES
(AC)	172	SIGNED	2	MRT1	INTERVAL T1 (TIME SLICE)
(AE)	174	SIGNED	2	MRT2	INTERVAL T2
(B0)	176	SIGNED	2	MRT3	INTERVAL T3 (POLLING TIME)
(B2)	178	SIGNED	2	MRT4	INTERVAL T4
(B4)	180	BITSTRING	2	MRSMSK	SHARED SPOOLING SYSID MASK
(B6)	182	BITSTRING	2	MRSNEG	COMPLEMENT SYSID MASK
(B8)	184	BITSTRING	4		RESERVED
(BC)	188	BITSTRING	1	MRTFLG	TIMER TASK FLAG
		1...		MRTFDLIM	"X'80'" .. RECALCULATE QCDLIM WHEN DFILE EXTENSION FINISHED
(BD)	189	BITSTRING	1	MRSY	SYS-ID OF OUR SYSTEM
(BE)	190	BITSTRING	1	MRSO	SHARED SPOOLING OPTION BYTE
(BF)	191	BITSTRING	1	MRSO2	SHARED SPOOL. OPTION BYTE 2
		1...		MRSO2T5	"X'80'" .. WAIT T5 OPTION FLAG
	GENERATION DEFAULTS & SETTINGS				
(C0)	192	SIGNED	4	QCDLIM	DBLK GROUP LIMIT NUMBER
(C4)	196	SIGNED	4	QCQLIM	QUEUE FILE LIMIT VALUE
(C8)	200	ADDRESS	4	QCTIME	TIME WHEN MSG LAST ISSUED
(CC)	204	CHAR-ACTER	8	QCMT	MEMBER TYPE DEFAULT
(D4)	212	CHAR-ACTER	4	QCJECL	ALTERNATE JECL PREFIX
(D8)	216	SIGNED	2	QCOEXWA	OUTPUT EXIT WORK AREA SIZE
(DA)	218	SIGNED	2	QCREXWA	READER EXIT WORK AREA SIZE
(DC)	220	SIGNED	4	QCQUSES	MAX Q-REC USED IN SESSION
(E0)	224	CHAR-ACTER	8	QCMPWD	MASTER PASSWORD

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(E8)	232	CHARACTER	8	MRSECN	LOCAL SECURITY NODEID
(F0)	240	CHARACTER	1	MRSECAC	SECURITY ACCESS CONTROL MODE
		11.. 1...		MRSECOY	C"Y"=SPOOL ACCESS PROT.ACTIVE
		11.1 .1.1		MRSECON	"C'N'" WHERE 'N' MEANS SPOOL ACCESS PROTECT
(F1)	241	BITSTRING	3		UNUSED
(F4)	244	ADDRESS	4	QCDBUSES	MAX. DBLK-GP'S IN SESS.
(F8)	248	BITSTRING	8		UNUSED
(100)	256	BITSTRING	128		RESERVED

AUXILIARY QUEUE RECORD AREA

Auxiliary queue record area. This area is required as work space for an additional queue record, used by the various queue management functions. The body fields of the queue record contain information pertinent to this particular queue entry and the user job which created it. The control fields of the queue record contain information relating to the status of the queue record and to its position within the VSE/POWER queues. See also the description of the Queue Record Area (QRA).

(180)	384	CHARACTER	368	QCQR (0)	AUXILIARY QUEUE REC. AREA
(180)	384	CHARACTER	256	QCPT1 (0)	QUEUE RECORD PART 1
(180)	384	CHARACTER	136	QCBF (0)	BODY FIELDS
(180)	384	CHARACTER	8	QCDY	DATE (CREATING SYST. FORMAT)
(188)	392	CHARACTER	35	QCSA (0)	INTERNAL REFERENCE FIELD
(188)	392	CHARACTER	4	QCST	OPERATION START TIME
(18C)	396	CHARACTER	4	QCET	OPERATION END TIME
(190)	400	CHARACTER	16	QCUI	USER INFORMATION
(1A0)	416	CHARACTER	8	QCNM	JOB NAME
(1A8)	424	SIGNED	2	QCJNO	JOB NUMBER
(1AA)	426	BITSTRING	1	QCQI	QUEUE RECORD IDENTIFIER
(1AB)	427	BITSTRING	1	QCCN	VSE/POWER CANCEL CODE
(1AC)	428	BITSTRING	1	QCRJ	LINE IDENTIFIER
(1AC)	428			QCDT	"QCRJ" DEVICE TYPE
(1AD)	429	CHARACTER	3	QCCU	CHANNEL AND UNIT (LINE ADDRESS)
(1B0)	432	BITSTRING	1	QCFJ	FROM TERMINAL IDENTIFIER
(1B1)	433	BITSTRING	1	QCTJ	TO TERMINAL IDENTIFIER
(1B2)	434	CHARACTER	1	QCCL	CLASS
(1B3)	435	CHARACTER	1	QCPY	PRIORITY
(1B4)	436	SIGNED	4	QCNR	RECORD COUNT
(1B8)	440	BITSTRING	1	QCPYSL	PRIORITY - SAVED LOCAL
(1B9)	441	BITSTRING	1	QCUEX	USER EXIT WORK BYTE, MUST NOT BE USED BY POWER.
(1BA)	442	BITSTRING	1	QCSN	JOB SUFFIX NUMBER
		1...		QCSNLA	"X'80'" .. LAST SEGMENT INDICATOR
(1BB)	443	SIGNED	1	QCNC	NUMBER OF COPIES
(1BC)	444	CHARACTER	4	QCFI	FORMS IDENTIFIER
(1C0)	448	SIGNED	4	QCCREC	CHECKPOINT RECORD NUMBER
(1C4)	452	CHARACTER	2	QCDYC	CENTURY OF CREATION DATE
(1C6)	454	ADDRESS	1	QCCCPY	CHECKPOINT COPY NUMBER
(1C7)	455	BITSTRING	1	QCDGP0	DUE DATE GENERAL BYTE 0
		1...		QCDG0X	"X'80'" DUE DATE INFO EXISTS
		.1..		QCDG0W	"X'40'" ENTRY QUEUED IN WFR-SQ
(1C8)	456	SIGNED	4	QCLC	LINE/CARD COUNTER

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(1CC)	460	SIGNED	4	QCRR	RESTART PAGE COUNT
(1D0)	464	SIGNED	1	QCCR	COPIES REMAINING
(1D1)	465	CHARACTER	1	QCDI	NEW DISP OR PURGE/FLUSH IND
(1D2)	466	CHARACTER	1	QCDP	DISPOSITION
(1D3)	467	SIGNED	1	QCSP	NUMBER OF SEPARATORS
(1D4)	468	SIGNED	4	QCBS	NUMBER OF RECORDS BEFORE SPLIT
(1D8)	472	SIGNED	4	QCBM	MAXIMUM VALUE OF COUNT
(1DC)	476	SIGNED	4	QCBN	ADDITIONAL COUNT VALUE
(1E0)	480	BITSTRING	2	QCER	3540 UNIT SPECIFICATION

FOR OUTPUT QUEUE ENTRIES FROM XW, THE ABOVE
FIELD IS USED TO SAVE THE PAGE LENGTH.

(1E2)	482	SIGNED	2	QCJ#	SAVE JOB NUMBER FOR ACCNT
(1E4)	484	CHARACTER	4	QCCP	COMPACTION TABLE NAME

3800 PRINTER CONTROL INFORMATION
OVERLAYED BY DUE DATE INFO (ONLY FOR RDR POSSIBLE)

(1E8)	488	CHARACTER	4	QCFL	FORMS OVERLAY IDENTIFIER
(1EC)	492	BITSTRING	8	QCCG	COPY GROUPS
(1F4)	500	BITSTRING	1	QCTC	TRANSMISSION COUNT
(1F5)	501	BITSTRING	1	QCCI	COPY GROUP INDEX
(1F6)	502	BITSTRING	1	QCPS	PAPER STATUS

CONTINUATION OF GENERAL SECTION

(1F7)	503	BITSTRING	1	QCOP	GENERAL OPTION BYTE 1
(1F8)	504	CHARACTER	8	QCPW	PASSWORD
(200)	512	ADDRESS	2	QCOJ#	ORIGINAL JOB NUMBER
(202)	514	CHARACTER	1	QCSID	SYSID OF TARGET CPU
(203)	515	CHARACTER	1	QCODP	ORIGINAL DISPOSITION
(204)	516	ADDRESS	2	QCRL	MAX RECORD LENGTH
(206)	518	BITSTRING	1	QCRCFM	RECORD FORMAT
(207)	519	BITSTRING	1	QCVOL	Q-ENTRY LABELED TAPE FLAG
		1...		QCVLAST	"X'80'" .. LAST MULTI-VOLUME "X'7F'" .. (VOLUME NUMBER) THE MAX VOLUME NUMBER IS 126. ANY VALUE OVER 126 MEANS GREATER OR EQUAL 127.

CONTROL SECTION
THE CONTROL PORTION OF THE QUEUE RECORD CONTAINS INFORMATION
RELATING TO THE STATUS OF THE QUEUE RECORD AND TO ITS
POSITION WITHIN THE VSE/POWER QUEUES.
NOTE: POFFLOAD LOAD/SELECT WILL COPY CERTAIN BYTES OF
THIS SECTION. OTHER BYTES ARE NOT MAINTAINED.

(208)	520	CHARACTER	48	QCCF (0)	CONTROL FIELDS
(208)	520	CHARACTER	1	QCXS	EXECUTION SWITCH C'X' ..ENTRY BEING PROCESSED
(209)	521	BITSTRING	1		RESERVED
(20A)	522	BITSTRING	1	QCRX	RESTART FUNCTION INDEX
(20B)	523	BITSTRING	1	QCSY	SYSTEM ID PROCESSING QR
(20C)	524	BITSTRING	1	QCS1	CONTROL FLAG BYTE 1
(20D)	525	BITSTRING	1	QCS2	CONTROL FLAG BYTE 2
(20E)	526	BITSTRING	1	QCS3	CONTROL FLAG BYTE 3
(20F)	527	BITSTRING	1	QCACN1	NON SHARED BROWSE COUNT OR SHARED SYS 1+2 BROWSE COUNT
(210)	528	ADDRESS	4	QCCRCT	PUT CHECKPOINT REC NUMBER
(214)	532	ADDRESS	4	QCRBC	CARDS/PAGES BEFORE CHKPT

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(218)	536	BITSTRING	4	QCAC39	SHARED SYSID 3-9 BROWSE CNT.
(21C)	540	BITSTRING	8	QCADD	ADD 'STCK' STAMP
(224)	548	ADDRESS	4	QCQP	PREVIOUS SET IN QUEUE
(228)	552	ADDRESS	4	QCQN	NEXT SET IN QUEUE
(22C)	556	ADDRESS	4	QCDF	1ST DBLK NO OF 1ST DBLK GP
(230)	560	ADDRESS	4	QCLDF	1ST DBLK NO OF LAST DBLK GP
(234)	564	ADDRESS	4	QCNB	NO OF DBLK GROUPS USED
EXTENSION OF THE BODY FIELDS					
(238)	568	CHARACTER	72	QCB2 (0)	BODY FIELDS EXTENSION
(238)	568	CHARACTER	8	QCTN	TARGET NODE NAME
(240)	576	CHARACTER	8	QCTU	TARGET USER ID
(248)	584	CHARACTER	8	QCON	ORIGINATOR NODE NAME
(250)	592	CHARACTER	8	QCOU	ORIGINATOR USER NAME
(258)	600	ADDRESS	4	QCWFRN	PTR TO NEXT WFR SUBQ ENTRY
OVERLAY SECTION 1 (DIFFERENT USAGE FOR INPUT AND OUTPUT QUEUE ENTRIES) USED FOR OUTPUT QUEUE ENTRIES					
(25C)	604	CHARACTER	8	QCOUT1 (0)	OUTPUT RELATED FIELD
(25C)	604	CHARACTER	8	QCDIST	DISTRIBUTION CODE
USED FOR INPUT QUEUE ENTRIES					
(25C)	604	CHARACTER	8	QCSECN	SECURITY NODEID
CONTINUATION OF GENERAL SECTION					
(264)	612	BITSTRING	1	QCOP2	GENERAL OPTION BYTE 2
		1...		QCO2BT	"X'80'" IGNORE BLANK TRUNCATION
		.1..		QCO2MSG	"X'40'" ISSUE MESSAGE 1Q4DI
		..1.		QCO2LGNO	"X'20'" LOG=NO SPECIFIED
		...1		QCO2XXXX	"X'10'" UNUSED
		.... 1...		QCO2QCM	"X'08'" QUEUE COMPLETION MESSAGE
		.... .1..		QCO2MR	"X'04'" GCM R-MSG FOR PRELEASE
		.... ..1.		QCO2MQ	"X'02'" GCM R-MSG ACC. TO Q-RECORD
(265)	613	BITSTRING	1	QCFLGO	FLAG BYTE FOR IN- & OUTPUT
		1...		QCCKI	"X'80'" .. CKP INFO EXISTS
		.1..		QCCKE	"X'40'" .. CKP INFO NOT AVAILABLE
		..1.		QCSAN	"X'20'" .. NOT SPOOL ACCESS PROTECTD
OVERLAY SECTION 2 (DIFFERENT USAGE FOR INPUT AND OUTPUT QUEUE ENTRIES) USED FOR OUTPUT QUEUE ENTRIES					
(266)	614	CHARACTER	18	QCOUT (0)	OUTPUT RELATED FIELDS
(266)	614	BITSTRING	1	QCOTF1	OUTPUT FLAG BYTE 1
		1...		QCOF1X80	"X'80'" UNUSED
		.1..		QCOF1X40	"X'40'" UNUSED
		..1.		QCOF1X20	"X'20'" UNUSED
		...1		QCOF1X10	"X'10'" UNUSED
		.... 1...		QCOF1LM	"X'08'" LINE-MODE STATE
		.... .1..		QCOF1LMI	"X'04'" LINE-MODE-IDM/IMM STATE
		.... ..1.		QCOF1PM	"X'02'" PAGE-MODE STATE
		1		QCOF1PM8	"X'01'" PAGE-MODE STATE
(267)	615	BITSTRING	1		RESERVED FOR FUTURE USE
(268)	616	SIGNED	4	QCPGN	PAGE COUNT
(26C)	620	SIGNED	2	QCRLLM	PRESERVE SPLDLREC PUT-APPEND
(26E)	622	SIGNED	2		RESERVED FOR FUTURE USE

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(270)	624	BITSTRING	8		RESERVED FOR FUTURE USE

USED FOR INPUT QUEUE ENTRIES
USE 'DS' INSTRUCTION TO KEEP DEFAULT VALUES
OF ABOVE DEFINITIONS

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(266)	614	CHARACTER	18	QCINP (0)	INPUT RELATED FIELDS
(266)	614	BITSTRING	1		RESERVED FOR FUTURE USE
(267)	615	BITSTRING	17	QCMRIN (0)	GCM R-MSG FOR PRELEASE
(267)	615	BITSTRING	1	QCMRSI	SYSID FOR GCM R-MSG
(268)	616	CHARACTER	8	QCMRAP	APPL FOR GCM R-MSG
(270)	624	CHARACTER	8	QCMRUS	USER FOR GCM R-MSG

CONTINUATION OF GENERAL SECTION

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(278)	632	CHARACTER	1	QCTDP	TRANSMISSION DISPOSITION
(279)	633	BITSTRING	7		RESERVED FOR FUTURE USE

Q-RECORD PART 2

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(280)	640	CHARACTER	112	QCPT2 (0)	QUEUE RECORD PART 2

EXTENSION -A OF THE CONTROL SECTION
X'100' --> CONTROL SECTION PART 2 --> X'11F'
CLEARED BY IPW$RQS
THE CONTROL PORTION OF THE QUEUE RECORD CONTAINS INFORMATION
RELATING TO THE STATUS OF THE QUEUE RECORD AND TO ITS
POSITION WITHIN THE VSE/POWER QUEUES.

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(280)	640	CHARACTER	32	QCC2 (0)	CONTROL FIELDS EXTENSION-A

RESTART TO ACTIVE RECORD CONTROL AREA

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(280)	640	ADDRESS	4	QCOTC	ADDRESS OF OWNING TCB OF UPDATE OR CREATE TASK
(284)	644	BITSTRING	12	QCCC (0)	CUR. RECORD COUNTS MAINT'ED BY $$GD FOR UPDATE/BROWSE BY $$PD FOR CREATE TASK
(284)	644	BITSTRING	4	QCCCNR	INTERNAL RECORD COUNT
(288)	648	BITSTRING	4	QCCCLC	DATA RECORD COUNT
(28C)	652	BITSTRING	4	QCCCPG	PAGE COUNT (USED BY $$PD)
(290)	656	BITSTRING	16		RESERVED FOR FUTURE USE

EXTENSION -B OF THE BODY FIELDS
X'120' --> BODY FIELDS PART 3 --> X'16F'

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(2A0)	672	CHARACTER	80	QCB3 (0)	BODY FIELDS EXTENSION -B
(2A0)	672	BITSTRING	80		RESERVED FOR FUTURE USE
(2A0)	672		0	QCEND	"*" END OF QUEUE RECORD

DUE DATE INFORMATION
OVERLAYS: 3800 PRINTER CONTROL INFORMATION

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(1E8)	488	CHARACTER	15	QCDD (0)	START OF INFO
(1E8)	488	BITSTRING	1	QCDGP1	GENERAL PURPOSE BYTE 1
		1...		QCDG1R	"X'80'" RERUN=NO SPECIFIED
		.1..			"X'40'" RESERVED
		..1.			"X'20'" RESERVED
		...1		QCDG1F	"X'10'" DUEFRQ SPECIFIED
		.... 1...		QCDG1T	"X'08'" DAILY SPECIFIED
		.... .1..		QCDG1W	"X'04'" WEEKDAYS SPECIFIED
		.... ..1.		QCDG1D	"X'02'" DAYS WITHIN MONTH
		1		QCDG1M	"X'01'" MONTHS SPECIFIED
		.... 111.		QCDG1C	"QCDG1T+QCDG1W+QCDG1D" CYCLING INFO ?
(1E9)	489	BITSTRING	1	QCDGP2	GENERAL PURPOSE BYTE 2

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
		1...		QCDG2F	"X'80'" 1ST TIME, NO NUMBER CHANGE
(1EA)	490	CHARACTER	6	QCDCY (0)	START OF CYCLING INFO
(1EA)	490	BITSTRING	2	QCDMY	MONTHS WITHIN YEAR LEFT ALIGNED: 80=JAN, 40=FEB, 20=MAR, ...
(1EC)	492	BITSTRING	4	QCDDM	DAYS WITHIN MONTH LEFT ALIGNED: 80=1ST, 40=2ND, 20=3RD, ...
(1F0)	496	CHARACTER	6	QCDN (0)	START OF NEXT DUE DATE PACKED DECIMAL WITHOUT SIGN
(1F0)	496	CHARACTER	4	QCDNDT (0)	NEXT DUE DATE (W/O TIME)
(1F0)	496	BITSTRING	2	QCDNY	YEAR (1988-2087)
(1F2)	498	BITSTRING	1	QCDNM	MONTH (1-12)
(1F3)	499	BITSTRING	1	QCDND	DAY (1-31)
(1F4)	500	CHARACTER	2	QCDNT (0)	START OF NEXT DUE TIME
(1F4)	500	BITSTRING	1	QCDNTH	HOUR (0-23)
(1F5)	501	BITSTRING	1	QCDNTM	MINUTES (0-59)
(1F6)	502	BITSTRING	1	QCDFQM	MINUTES (0-59) OF DUEFRQ
		 1111		QCDLEN	"*-QCDD" LENGTH OF DUE DATE
(1EC)	492	BITSTRING	1	QCDDW	WEEKDAYS
		1...		QCDWMO	"X'80'" MONDAY
		.1..		QCDWTU	"X'40'" TUESDAY
		..1.		QCDWWE	"X'20'" WEDNESDAY
		...1		QCDWTH	"X'10'" THURSDAY
		.... 1...		QCDWFR	"X'08'" FRIDAY
		.... .1..		QCDWSA	"X'04'" SATURDAY
		.... ..1.		QCDWSU	"X'02'" SUNDAY
(1EA)	490	CHARACTER	2	QCDFT (0)	START OF FIRST TIME
(1EA)	490	BITSTRING	1	QCDFTH	HOUR (0-23)
(1EB)	491	BITSTRING	1	QCDFTM	MINUTE (0-59)
(1EC)	492	BITSTRING	1		USED FOR WEEKDAYS
(1ED)	493	CHARACTER	2	QCDLT (0)	START OF LAST TIME
(1ED)	493	BITSTRING	1	QCDLTH	HOUR (0-23)
(1EE)	494	BITSTRING	1	QCDLTM	MINUTE (0-59)
(1EF)	495	BITSTRING	1	QCDFQH	HOURS (0-23) OF DUEFRQ
SOME DISPLACEMENTS FOR THE OLD VERSION OF THE QUEUE RECORD, I.E. VERSION 5.1 AND PREVIOUS ONES					
		.1.. .1..		QCOVNP	"X'44'" PAGE NO, 2 BYTES ONLY
FIRST PART OF SPOOL ENVIRONMENT RECORD					
(2F0)	752	BITSTRING	16	QCSER	SPOOL ENVIRONMENT RECORD
(300)	768	BITSTRING	16		RESERVED FOR SER EXTENSION
(310)	784	BITSTRING	16		UNUSED
(310)	784		0	QCFLN	"(*-QCSD)" LENGTH OF FIRST PART OF DMB
MASTER RECORD AREA					
The master record is written as the last physical record within the queue file extent. During VSE/POWER execution a copy of the master record is maintained in this area. Whenever this copy is updated a replacement master record is at once written to the queue file so that, in the event of a failure of the system, warm start information can be recovered from the direct access device in question.					
(320)	800	SIGNED	4	QCMR (0)	MASTER RECORD AREA
(320)	800	CHARACTER	4	MRVM	QUEUE VERSION LEVEL
(324)	804	SIGNED	2	MRNO	MASTER JOB NUMBER
(326)	806	SIGNED	2	MRUC	USE-COUNT
(328)	808	BITSTRING	8	MRCODY	DATE (CURR. SYS. FORMAT) OF LAST VSE/POWER COLD START
QUEUE FILE INFORMATION					
(330)	816	SIGNED	4	MRQECB	EVENT CONTROL BLOCK
(334)	820	ADDRESS	4	MRQFRNO	FIRST RECORD IN FREE QUEUE

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(338)	824	SIGNED	4	MRQ#MAX	TOTAL NO OF USABLE QUEUE REC
		.... ..1.		NRQRO	"2" NO OF OVERHEAD QUEUE REC'S
(33C)	828	SIGNED	4	MRQFREE	NR OF FREE QUEUE RECORDS
(340)	832	SIGNED	4	MRQUSED	MAXIMUM NO OF QUEUE REC USED
(344)	836	SIGNED	4	MRQRBAD	NUMBER OF BAD QUEUE RECORDS
(348)	840	SIGNED	2	MRQRBLK	TOTAL NO OF QUEUE REC BLOCKS
(34A)	842	SIGNED	2	MRQRBLN	QUEUE RECORD BLOCK SIZE
(34C)	844	SIGNED	2	MRQRRCN	NO QUEUE RECORDS PER BLOCK
(34E)	846	SIGNED	2	MRQRCSZ	QUEUE REC COMPARTMENT SIZE
(350)	848	BITSTRING	4	MRQRDEL	NUMBER OF DELAYED Q-RECORDS
POFFLOAD PICKUP FLAG (SHARED OR NON-SHARED QFILE)					
(354)	852	BITSTRING	1	MRPKUP	POFFLOAD PICKUP RUNNING IND X'40'- RUNNING NON-SHARED C'N' - RUNNING SHARED(SYSID)
(355)	853	BITSTRING	3		UNUSED
(358)	856	BITSTRING	8		UNUSED
DATA FILE INFORMATION					
(360)	864	SIGNED	4	MRDB	DATA BLOCK SIZE (DBLK)
(364)	868	SIGNED	4	MRDBGP	DBLK GROUP SIZE
(368)	872	SIGNED	4	MRDBMAX	NUMBER OF TOTAL DBLK GROUPS
(36C)	876	SIGNED	4	MRDBFRE	NUMBER OF FREE DBLK GROUPS
(370)	880	SIGNED	4	MRDBUSE	MAX NO OF DBLK GROUPS USED
(374)	884	SIGNED	4	MRDBBAD	NUMBER OF BAD DBLK GROUPS
(378)	888	SIGNED	4	MRDECB	EVENT CONTROL BLOCK
(37C)	892	BITSTRING	1	MR#E	NO OF DATAFILE EXTENTS
(37D)	893	BITSTRING	1	MRDFLAG	DATA FILE FLAG BYTE
		1...		MRDBLKTR	"X'80'" .. DBLKGP TRACE ENABLED
		.1..		MRDF2BIL	"X'40'" .. MORE 2,147,483,647 DBLKS
		..1.		MRDFEXTP	"X'20'" .. DFILE EXTENSION PLANNED
		...1		MRDFEXTF	"X'10'" .. DFILE EXTENSION FAILED
(37E)	894	BITSTRING	1	MRDEXTSY	SYSID OF ADD. DFILE EXTENT FORMATTING SYSTEM
(37F)	895	BITSTRING	1	MRDFPUBC	DATA FILE PUB DEV.TYPE CODE
(380)	896	BITSTRING	4	MRDBDEL	NUMBER OF DELAYED DBLKGP'S
(384)	900	SIGNED	4	MRDBEMAX	EXT. NUMBER OF TOTAL DBLKGPS
EXTENT SPECIFICATION TABLE THE FOLLOWING TABLE DESCRIBES EACH DATA FILE EXTENT BY ITS LOGICAL UNIT, START POINT (CKD TRACKS|FBA BLKS) AND LENGTH (CKD TRACKS|FBA BLKS). THIS TABLE IS SET IN IPW$$I4 AT DX20/DXF10 FOR COLD START OR EXTENSION WARMSTART.					
(388)	904	BITSTRING	9	MRDFEXT (32)	32 SLOTS, 9 BYTE PER EXTENT LOG.UNIT, START & LENGTH
FREE DBLK GROUP SUBCHAINS THE FOLLOWING TABLE CONSISTS OF 8 ENTRIES. EACH ENTRY DESCRIBES A FREE DBLK GROUP SUBCHAIN.					
(4A8)	1192	SIGNED	4	MR$1NO	ADDR OF 1ST FREE DBLK GROUP
(4AC)	1196	SIGNED	4	MR$1CT	NUMBER OF FREE DBLK GROUPS
		.... 1...		MR$1LEN	"*-MR$1NO" LENGTH OF ONE ENTRY
(4B0)	1200	BITSTRING	8	(7)	ENTRIES 2 - 8
		.... 1...		MRDBSUB	"(*-MR$1NO)/MR$1LEN" NUMBER OF SUBCHAINS
(4E8)	1256	SIGNED	1	MRFMT#E	NO. FORMATTED DFILE EXT'S
(4E9)	1257	BITSTRING	15		UNUSED
QUEUE CONTROL AREA INFORMATION USED BY SLOT MANAGER					
(4F8)	1272	ADDRESS	2	QCASNGP	NO OF DBLK GROUPS USED
(4FA)	1274	SIGNED	2	QCASNWS	NO OF WAITING FOR WORK SLOTS
(4FC)	1276	BITSTRING	1	QCASSWI	SWITCH BYTE
		.1..		QCASSEX	"X'40'" ..QCA PRESENT
(4FD)	1277	BITSTRING	1		UNUSED
(4FE)	1278	BITSTRING	2		UNUSED
(500)	1280	ADDRESS	4	QCASDSA	REL. NUMBER OF 1ST DBLK

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(504)	1284	BITSTRING	4	QCASSYS (0)	CNTL INFO FOR ONE SYSTEM
(504)	1284	ADDRESS	2	QCASNOS	NO OF ACTIVE SLOTS/SYSID
(506)	1286	BITSTRING	1	QCASFLG	FLAG BYTE
		1...		QCASFAC	"X'80'" ..1ST SLOT-DBLK MODIFIED
(507)	1287	BITSTRING	1		RESERVED FOR FUTURE USE
(508)	1288	BITSTRING	4	(9)	ENTRIES FOR SYSID 1 - 9
(52C)	1324	BITSTRING	4		SYSID = 10 FOR CKP SLOTS
		.... 1.11		QCASNOE	"(*-QCASSYS)/L'QCASSYS" NO OF ENTRIES OF QCASSYS

ACCOUNT FILE VALUES

(530)	1328	BITSTRING	8	MRAS	LAST RECORD ADDRESS IJAFILE
(538)	1336	BITSTRING	2	MRSE	SECTOR VALUES
(53A)	1338	BITSTRING	2		RESERVED FOR FUTURE USE
(53C)	1340	SIGNED	4	MRCF	FREE SPACE IN CURRENT CI OR RESIDUAL CAP CURRENT TRACK
(540)	1344	SIGNED	4	MRCC	CURRENT RESIDUAL CAPACITY
(544)	1348	BITSTRING	12		UNUSED

MASTER CLASS TABLE AREA

THE FOLLOWING PART OF THE CONTROL BLOCK CONTAINS THE THREE
SYMMETRICAL CLASS TABLES WHICH DEFINE THE STATUS OF THE
VSE/POWER QUEUES. ONE TABLE IS MAINTAINED FOR READER
INPUT CLASSES, ONE FOR LIST OUTPUT CLASSES, AND ONE FOR
PUNCH OUTPUT CLASSES. EACH TABLE COMPRISES 37 ENTRIES,
WHERE ENTRY REPRESENTS A FOUR BYTE 'FIRST-IN-QUEUE'
AND A FOUR BYTE 'LAST-IN-QUEUE' RECORD NUMBER FOR THE
FOLLOWING CLASSES:
X'FA' -> 1 INTERNAL USE, E.G. $SPLNNNN DISP. ENTRY
0 - 9 -> 10 NUMERIC CLASSES
A - Z -> 26 ALPHABETIC CLASSES
NOTE, THAT THE LEFTMOST (HIGH ORDER BIT) OF THE 'LAST-
IN-QUEUE' RECORD NUMBER IS USED AS 'ADD-TO-CLASS'
POST BIT, WHERE TASKS WAIT UPON BY THE IPW$WFQ MACRO.
THE XMIT QUEUE AREA WHICH FOLLOWS THE READER/LIST/PUNCH
CLASS AREA CONSISTS OF 2 ENTRIES, ONE FOR READER QUEUE
ENTRIES, AND ONE FOR OUTPUT (LIST/PUNCH) QUEUE ENTRIES.

(550)	1360	CHARACTER	888	QCCT (0)	RDR/LST/PUN CLASS ANCHORS, WITHOUT THE 2 XMT ANCHORS
(550)	1360	CHARACTER	296	CTRT (0)	RDR CLASS AREA
(550)	1360	BITSTRING	88		RDR CLASS AREA - PART 1
(5A8)	1448	BITSTRING	208		RDR CLASS AREA - PART 2
(678)	1656	CHARACTER	296	CTLT (0)	LST CLASS AREA
(678)	1656	BITSTRING	88		LST CLASS AREA - PART 1
(6D0)	1744	BITSTRING	208		LST CLASS AREA - PART 2
(7A0)	1952	CHARACTER	296	CTPT (0)	PUN CLASS AREA
(7A0)	1952	BITSTRING	88		PUN CLASS AREA - PART 1
(7F8)	2040	BITSTRING	208		PUN CLASS AREA - PART 2
(8C8)	2248	BITSTRING	1	CTXT	XMT QUEUE CLASS ANCHORS 1 ANCHOR FOR RDR ENTRIES 1 ANCHOR FOR LST & PUN ENTR.
(8C8)	2248		0	QCCTLN	"*-QCCT" LENGTH OF DISP. CLASS TABLE

THE FOLLOWING TABLES CONTAIN THE CLASS ANCHORS FOR
NON-DISPATCHABLE QUEUE ENTRIES IN THE RDR, LST, PUN AND
XMT QUEUE. THE TABLES ARE SYMMETRICAL TO THE DISPATCHABLE
CLASS TABLES.

(8D8)	2264	BITSTRING	88	QCCTNDP	NON-DISP. RDR QUEUE, PART 1
(930)	2352	BITSTRING	208		NON-DISP. RDR QUEUE, PART 2
(A00)	2560	BITSTRING	88		NON-DISP. LST QUEUE, PART 1
(A58)	2648	BITSTRING	208		NON-DISP. LST QUEUE, PART 2
(B28)	2856	BITSTRING	88		NON-DISP. PUN QUEUE, PART 1
(B80)	2944	BITSTRING	208		NON-DISP. PUN QUEUE, PART 2
(C50)	3152	BITSTRING	1		NON-DISP. XMT QUEUE

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
WAIT FOR RUN SUBQUEUE INFO					
(C60)	3168	ADDRESS	4	MRWFRPT	PTR TO 1ST WFR SUBQUEUE ENT
(C64)	3172	BITSTRING	2	MRWFRRF	REFRESH BITS FOR SYSID 1-9
(C66)	3174	BITSTRING	2		UNUSED
SHARED SPOOLING CONTROL INFORMATION					
(C68)	3176	BITSTRING	1	SSID	SYSTEM-ID OWNING Q-FILE
(C69)	3177	BITSTRING	1	SSF1	FLAG BYTE 1
		1...		SSQS	"X'80'" .. QUEUE FILE SHARED
		.1..		SSAS	"X'40'" .. ACCOUNT FILE SHARED
(C6A)	3178	BITSTRING	1	SSF2	FLAG BYTE 2
(C6B)	3179	BITSTRING	1	SSAC	SYSTEM-ID OWNING ACCT FILE
(C6C)	3180	BITSTRING	4	SSWK	WORK TO DO ECB
(C70)	3184	BITSTRING	4	SSNW	NO-WORK TO DO ECB
(C74)	3188	BITSTRING	10	SSCT	SYS-ID BUCKET
(C7E)	3198	BITSTRING	80	SSCPT	CPU-ID BUCKET
(CCE)	3278	BITSTRING	1	SSMSW	MASTER SWITCH
		1...		SSMQD	"X'80'" .. QUEUE FILE DAMAGED
		.1..		SSMUC	"X'40'" .. USE CORE COPY OF Q-FILE
(CCF)	3279	BITSTRING	9		UNUSED
SHARED REMOTE-ID TABLE THE FOLLOWING TABLE INDICATES FOR EACH REMID OF A SHARED CONFIGURATION IF OUTPUT IS AVAILABLE. AN INDICATION IS SET IF THE REQUIRED REMOTE TERMINAL IS NOT ATTACHED TO THE SYSTEM THAT PRODUCED THE OUTPUT. FOR EACH REMID STANDS A SINGLE BIT, WHICH IS SET TO INDICATE THAT AT LEAST ONE OUTPUT ENTRY IS ELIGIABLE TO BE PROCESSED BY THE SYSTEM WITH THE GIVEN REMOTE TERMINAL.					
(CD8)	3288	BITSTRING	32	SSRT	REMOTE TABLE
SYSID CLASS TABLE THE FOLLOWING TABLE INDICATES FOR EACH SYSID IF JOBS ARE IN ONE OF THE VSE/POWER QUEUES. EACH CLASS IN THE QUEUE IS REPRESENTED BY A BIT. IF THE BIT IS ON, IT INDICATES THAT AT LEAST ONE JOB IS IN THE CLASS QUEUE, ELIGIABLE TO BE PROCESSED BY THE SYSTEM FOR THIS PARTICULAR SYSID. ON THE OTHER SIDE THE 'GET NEXT QUEUE SET' FUNCTION TURNS OFF THE BIT, WHEN IT DETECTS THAT THERE IS NO JOB IN THE CLASS QUEUE, ELIGIABLE TO BE PROCESSED BY THE SYSTEM, FOR THE GIVEN SYSID.					
(CF8)	3320	CHARACTER	1	SSST	ENTRY FOR SYSID '1'
(CF9)	3321	BITSTRING	5	SSRS	RDR QUEUE ENTRY
(CFE)	3326	BITSTRING	5	SSLS	LST QUEUE ENTRY
(D03)	3331	BITSTRING	5	SSPS	PUN QUEUE ENTRY
		...1		SSSL	"*-SSST" LENGTH OF ONE ENTRY
(D08)	3336	CHARACTER	25		ENTRY FOR SYSID '2'
(D18)	3352	CHARACTER	25		ENTRY FOR SYSID '3'
(D28)	3368	CHARACTER	25		ENTRY FOR SYSID '4'
(D38)	3384	CHARACTER	25		ENTRY FOR SYSID '5'
(D48)	3400	CHARACTER	25		ENTRY FOR SYSID '6'
(D58)	3416	CHARACTER	25		ENTRY FOR SYSID '7'
(D68)	3432	CHARACTER	25		ENTRY FOR SYSID '8'
(D78)	3448	CHARACTER	25		ENTRY FOR SYSID '9'

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(D88)	3464	BITSTRING	8		UNUSED
SECURITY NODEID(S)					
(D90)	3472	CHARACTER	8	MRSECNNS	NON-SHR'D WARMSTART SECNODE
THE FOLLOWING TABLE INDICATES FOR EACH SYSID THE SECURITY NODEID (IF ANY) SPECIFIED FOR THE (SHARED) CPU IN THE POWER GENERATION MACRO.					
(D90)	3472		0	SSSECN	"*"
(D98)	3480	CHARACTER	8		SECURITY NODEID FOR SYSID'1'
(DA0)	3488	CHARACTER	8		SECURITY NODEID FOR SYSID'2'
(DA8)	3496	CHARACTER	8		SECURITY NODEID FOR SYSID'3'
(DB0)	3504	CHARACTER	8		SECURITY NODEID FOR SYSID'4'
(DB8)	3512	CHARACTER	8		SECURITY NODEID FOR SYSID'5'
(DC0)	3520	CHARACTER	8		SECURITY NODEID FOR SYSID'6'
(DC8)	3528	CHARACTER	8		SECURITY NODEID FOR SYSID'7'
(DD0)	3536	CHARACTER	8		SECURITY NODEID FOR SYSID'8'
(DD8)	3544	CHARACTER	8		SECURITY NODEID FOR SYSID'9'
MASTER RECORD SPOOL ACCESS PROTECTION STORE THE FOLLOWING TABLE INDICATES FOR A NON-SHARED SYSTEM THE SPOOL ACCESS PROTECTION MODE.					
(DE0)	3552	BITSTRING	1	MRSECACN	NON-SHR'D SPOOL ACCESS PROT MODE
THE FOLLOWING TABLE INDICATES FOR EACH SYSID THE SPOOL ACCESS PROTECTION MODE.F					
(DE1)	3553	BITSTRING	1	MRSECACS	SPOOL ACCESS PROT MODE SYSID'1'
(DE2)	3554	BITSTRING	1		SPOOL ACCESS PROT MODE SYSID'2'
(DE3)	3555	BITSTRING	1		SPOOL ACCESS PROT MODE SYSID'3'
(DE4)	3556	BITSTRING	1		SPOOL ACCESS PROT MODE SYSID'4'
(DE5)	3557	BITSTRING	1		SPOOL ACCESS PROT MODE SYSID'5'
(DE6)	3558	BITSTRING	1		SPOOL ACCESS PROT MODE SYSID'6'
(DE7)	3559	BITSTRING	1		SPOOL ACCESS PROT MODE SYSID'7'
(DE8)	3560	BITSTRING	1		SPOOL ACCESS PROT MODE SYSID'8'
(DE9)	3561	BITSTRING	1		SPOOL ACCESS PROT MODE SYSID'9'
(DEA)	3562	BITSTRING	6		UNUSED
DEFECT QUEUE RECORD BLOCK MAP					
(DF0)	3568	BITSTRING	1	MRDQBMAP (135)	DEFECT QUEUE REC BLOCK MAP, ONE BIT FOR EACH OF THE 3125 QUEUE RECORD BLOCKS - (NOTE 100,000 : 32 = 3125)
(F77)	3959	BITSTRING	1		UNUSED
NETWORKING CONTROL INFORMATION					
(F78)	3960	BITSTRING	8	MRNNEW	NEW LOCAL NODE NAME
(F80)	3968	CHARACTER	9	MROQ (0)	OWN NODE NAME AND QUALIFIER
(F80)	3968	CHARACTER	8	MRNN	NODE NAME OF OUR SYSTEM
(F88)	3976	BITSTRING	1	MRNNQ	NODE QUALIFIER
(F89)	3977	BITSTRING	7		UNUSED
GENERAL INFORMATION AND UNUSED AREA					
(F90)	3984	ADDRESS	4	MRRFOFF	OFFSET TO START REFRESH TBL.
(F94)	3988	BITSTRING	92		UNUSED

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
	NODE ATTACHED TABLE (NAT)				
(FF0)	4080	CHARACTER	800	MNAT (0)	80 ENTRIES NAT
(FF0)	4080	BITSTRING	200		EACH ENTRY HAS 10 BYTES
(10B8)	4280	BITSTRING	200		IN LENGTH
(1180)	4480	BITSTRING	200		
(1248)	4680	BITSTRING	1		
	REFRESH TABLE THE REMAINDER OF THE CONTROL BLOCK CONTAINS THE REFRESH TABLE. ALTHOUGH ONLY REQUIRED FOR A SHARED SPOOLING ENVIRONMENT, THE REFRESH TABLE IS ALWAYS APPENDED TO THE MASTER RECORD, SO THAT WARM START IS POSSIBLE BETWEEN NON-SHARED AND SHARED VSE/POWER SYSTEMS. THE TABLE CONSISTS OF 2-BYTES ENTRIES, ONE FOR EACH QUEUE RECORD BLOCK, PROVIDING SPACE FOR 3125 BLOCKS, WHICH ARE REQUIRED FOR A MAXIMUM QUEUE FILE OF 100,000 QUEUE RECORDS (32 RECORDS RESIDING IN A BLOCK). BY CORRESPONDING SYSID FLAG (1-9) BIT THE ENTRY INDICATES, IF THE APPROPRIATE QUEUE RECORD BLOCK MUST BE REFRESHED BY THE TIMER TASK (OF THE CORRESPONDING SYSID) BEFORE REFERENCING ITS QUEUE RECORDS.				
(1310)	4880	BITSTRING	1	MRRFTAB (106)	REFRESH TABLE
		.... ..1.		MRRFELN	"2" .. REFR. TABLE ENTRY LENGTH
(1310)	4880		0	MRULN	"(*-QCMR)" LENGTH OF USED MASTER REC.
(2B7A)	11130		0	MRTLN	"(*-QCMR)" LENGTH OF TOTAL MASTER REC., MUST NOT EXCEED 12288 BYTES I.E. X'3000' BYTES OF Q-BLK.

How to Locate: Refer to Figure 151 on page 730 in Chapter 6, “Diagnostic Aids.”

DSECTS for Accounting (A-FILE ON FBA)

Definition Macro: IPW$DJK

This macro maps 4 DSECTs:

- Sequential file header
- Control interval definition field (CIDF)
- Record definition field (RDF)
- I/O area for read device characteristics

The formats are as follows:

```
Bytes       Label
Hex.        of Field    Description/Function
----------------------------------------------------------------------

•  Sequential File Header

00-01       HEADBL      Block descriptor.  Contains length of
                        the physical block, including its own length.
02-03                   Reserved
04-05       HEADRL      Record descriptor.  Contains length of
                        the logical record, including its own length
06-07                   Reserved

•  Control Interval Definition Field

00-01       CIDFA       Start address of the free space in the
                        control interval
02-03       CIDFL       Length of free space in the control interval

•  Record Definition Field

00          RDFF        Flags (always zero)
01-02       RDFL        Contains the length of the corresponding record

•  I/O-Area for Read Device Characteristics

00                      Operation mode
01                      Features
02                      Device class
03                      Unit type
04-05       RDCBLOCK    Physical record size
06-09                   No. of blocks
0A-1F                   Reserved
```

Dynamic Partition Control Block

Definition macro: IPW$DEF DPCB=YES

The dynamic partition control block (DPCB) is built during VSE/POWER initialization by IPW$$I7, provided the system is operating in /370 or ESA mode. It is never released during VSE/POWER operation. Being the main control block of the dynamic partition scheduling task (DPST) it contains a list of up to ten pointers to the table of classes, for which the DPST will schedule jobs for execution in a dynamic partition. The DPCB also contains the address of the currently active dynamic class table (DCLT) located in the supervisor area.

Offset Hex	Type	Len	Name (Dim)	Description
(0)	CHARACTER	16	DPCBHD	SECTION DESCRIPTOR
(10)	BITSTRING	4		UNUSED
(14)	ADDRESS	4	DPCBECB	ECB USED BY DPST
(18)	ADDRESS	4	DPCBECBL	ECB USED WHEN DPCB LOCKED
(1C)	ADDRESS	4	DPCBLW	LOCK WORD
(20)	CHARACTER	48	DPCPTLST (0)	$WFM LIST OF ECB POINTERS
(20)	ADDRESS	4	DPCATCEB	TCEB ADDRESS OF DPST
(24)	ADDRESS	4	DPCACLAS (10)	UP TO TEN CLASS TABLE PTRS.
(4C)	ADDRESS	4	DPCAEND	LIST END INDICATOR
(50)	CHARACTER	44	DPCPSUAR (0)	POINTER SUSPENDED AREA
(50)	ADDRESS	4	DPCACLSU (11)	PTR. TO POSSIBLY SUSP. CLS.
(7C)	CHARACTER	44	DPCPREAR (0)	POINTER RE-ARRANGE AREA
(7C)	ADDRESS	4	DPCACLRA (11)	POINTERS TO CLASS TABLE
(A8)	ADDRESS	4	DPCBACT	ADDRESS OF ACTIVE DCLT
(AC)	ADDRESS	4	DPCBDPCE	ADDRESS OF DYN. PART. PCE
(B0)	SIGNED	4	DPCBSAL	SUCCESSFUL ALLOC. COUNT
(B4)	SIGNED	4	DPCBFAL	FAILING SPACE ALLOC. COUNT
(B8)	SIGNED	4	DPCBFNP	FAILING NO PARTITION COUNT
(BC)	SIGNED	4	DPCBFSG	FAILING SYSTEM GETV. COUNT
(C0)	SIGNED	4	DPCBFPR	FAILING POWER REAL COUNT
(C4)	SIGNED	4	DPCBFPV	FAILING POWER VIRT. COUNT
(C8)	BITSTRING	24	DPCBTAL	TIMER ELEMENT ALLOC MSG
(E0)	BITSTRING	24	DPCBTNP	TIMER ELEMENT NO PART. MSG
(F8)	BITSTRING	24	DPCBTSG	TIMER ELEMENT SYST. GETV.
(110)	BITSTRING	24	DPCBTPR	TIMER ELEMENT PWR REAL MSG
(128)	BITSTRING	24	DPCBTPV	TIMER ELEMENT PWR VIRT MSG
(140)	BITSTRING	24	DPCBTRS	TIMER ELEMENT FOR RESUME
(158)	BITSTRING	24	DPCBTPG	TIMER ELEMENT PFIXED S-GETV
(170)	BITSTRING	1	DPCBFL1	DPCB FLAG BYTE 1
	1...		DPC1ENDI	"X'80'" CLASS BEEN ENABLED/DISABLED
	.1..		DPC1RESU	"X'40'" RESUME SUSPENDED CLASS
	..1.		DPC1CHAN	"X'20'" ENABLED CLASS MIGHT CHANGE
	...1		DPC1CSFL	"X'10'" PSTART DYN. PARTITION FAILS
(171)	BITSTRING	3		UNUSED
(174)	BITSTRING	4		UNUSED
(178)	BITSTRING	24	DPCBTRO	TIMER ELEMENT REAL RUN OUT
(190)	BITSTRING	24	DPCBTVO	TIMER ELEMENT VIRT. RUN OUT
(1A8)	SIGNED	4	DPCBPUP	NUMBER PART. 'UP' TILL $$XW
	EXPRESSION		DPCBLN	"*-DPCBDS" LENGTH OF DPCB CTL. BLOCK

Execution Processor Work Area

Definition Macro: IPW$DEF XRWA=YES

The execution processor work area is reserved at execution reader/writer task initialization and used during validation of CCW and data address to hold the partition/system information retrieved via the EXTRACT macro.

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(0)	0	STRUCTURE		XRWDS	,
(0)	0	ADDRESS	4	XRWVPBA	VIRTUAL PART. BEGIN ADDR.
(4)	4	ADDRESS	4	XRWVEND	VIRT PART END (W/O GETVIS)
(8)	8	ADDRESS	4	XRWVPEA	VIRTUAL PART END ADDRESS
(C)	12	ADDRESS	4	XRWRPBA	REAL PART. BEGIN ADDRESS
(10)	16	ADDRESS	4	XRWRPEA	REAL PARTITION END ADDRESS
		...1 .1..		XRWPPALN	"*-XRWDS" LENGTH FOR EXTRACT MACRO
(14)	20	SIGNED	4	XRWEXPA (5)	EXTRACT MACRO PARM LIST
(28)	40	SIGNED	4	XRWSAV2	SAVE AREA R2
(2C)	44	SIGNED	4	XRWSAV3	SAVE AREA R3
(30)	48	BITSTRING	1	XRWFLG	FLAG BYTE
		1...		XRWFEX	"X'80'" .. EXTRACT DONE
		.1..		XRWFST	"X'40'" .. INITIAL ENTRY OF EX WRIT
		..1.		XRWSAN	"X'20'" .. ENTRY NOT SPOOL ACCESS PROTECTED
(31)	49	BITSTRING	3	XRWTKP	ADDR OF WAITING WRITER TASK
(34)	52	SIGNED	4	XRWTEMP	TEMP WORKAREA
(38)	56	CHARACTER	8	XRWSID	DEFAULT VSE SECURITY USERID
(40)	64	CHARACTER	8	XRWSPW	DEFAULT VSE SECURITY PASSWD
(48)	72	CHARACTER	8	XRWSECN	DEFAULT VSE SECURITY SECNODE
(50)	80	CHARACTER	8	XRWLTA	LTA PHASE NAME
(58)	88	CHARACTER	8		UNUSED
(60)	96	CHARACTER	8		UNUSED
(68)	104	CHARACTER	8		UNUSED
(70)	112	CHARACTER	8		UNUSED
		.111 1...		XRWLEN	"*-XRWDS" LENGTH OF WORK AREA

External Device Control Block (EDCB)

Definition Macro: IPW$DED

The external device control block (EDCB) is created by the VSE/POWER command processor when a PSTART for an external device is given. The control block contains DDS and device-related information used by the device service task. The EDCBs are chained off the master external device control block.

Bytes Hex.	Label of Field	Description/Function
00	EDCBDS	Start of DSECT
00-0F	EDCBHEAD	Storage descriptor
10-17	EDCBDDSN	Name of Device Driving System (DDS)
18-1B	EDCBATCB	Address of device service task TCB
1C-1F	EDCBNEXT	Address of next EDCB in chain
20-27	EDCBNAME	Device name
28-67	EDCBLOGN	Logical destination names, 8 bytes each
68-6B	EDCBCLSS	Class list
6C	EDCBSTAT	Status byte 1
	EDCBSTCC	X'80' - connection completed
	EDCBSTDS	X'40' - device started
	EDCBSTWW	X'20' - Device waiting for work
	EDCBSTWR	X'10' - Device waiting for reactivation
	EDCBSTSS	X'08' - 'Stop device order' sent
	EDCBSTSU	X'04' - 'Setup' in progress
	EDCBSTSR	X'02' - 'Device stopped' signal received
	EDCBSTSL	X'01' - 'Waiting for work' slot built
6D	EDCBSTA2	Status byte 2
	EDCBS2SQ	X'80' - 'output arrived' signal queued
	EDCBS2SO	X'40' - 'start device' order sent
	EDCBS2PO	X'20' - 'stop device' order sent
6E	EDCBTCOD	Termination code
	EDCBTTCE	X'80' - PSTOP EOJ given
	EDCBTTCI	X'40' - PSTOP IMM given
	EDCBTTCR	X'20' - PSTOP RESTART given
	EDCBTTCK	X'10' - PSTOP FORCE given
	EDCBTTCN	X'08' - PEND given
6F	EDCBQUID	Processing queue identifier (L or P)
70-73	EDCBAORD	Address of first order in queue, if any
74-77		Address of last order in queue, if any
78-7B	EDCBWRMS	Address of message, placing task operator bound
7C	EDCBORTP	Last sent order type
7D-87		Reserved for future use
88-9F	EDCBDSTU	Device owner information
88-8F	EDCBAPPL	Subsystem identifier of device owner
90-97	EDCBNODE	Node name of device owner
98-9F	EDCBUSER	User id of device owner
A0-B7	EDCBDSTC	Destination information of person who stopped device
A0-A7	EDCBSTAP	Subsystem name of person stopping device
A8-AF	EDCBSTNO	Node name of person stopping device
B0-B7	EDCBSTUA	User identifier of person stopping device

FCB Table (FCBCB)

Definition Macro: IPW$DEF FCTAB=YES

The FCB table is anchored in VSE/POWER's Control Address Table in field CAFCTAB. The table consists of two sections. The first section describes the table header containing the section descriptor, a register save area and the length of the . total FCB table including this first section. The second section provides FCB related data, such as name of FCB, its line table presentation, the printer page size and the length of the total table without section 1.

Offset Hex	Type	Len	Name (Dim)	Description
(0)	CHARACTER	16	FCBSD	SECTION DESCRIPTOR
(10)	SIGNED	4	FCBSAVR (4)	REGISTER SAVE AREA
	..1.		FCBHDRL	"*-FCBTDS" ..LENGTH OF HEADER
(20)	CHARACTER	1	FCBDUM	SPACE FOR 1 ENTRY
	...1 111.		FCBNUMT	"30" ..# OF TABLE ENTRIES
	...1 1.1.		FCBENTL	"L'FCBNAM+L'FCLTAB+L'FCPSIZ"..LENGTH OF ENTRY
	EXPRESSION		FCBTABL	"(FCNUMT*FCENTL)+FCBHDRL" ..LENGTH OF TOTAL TAB.

Offset Hex	Type	Len	Name (Dim)	Description
(0)	CHARACTER	8	FCBNAM	NAME OF FCB
(8)	CHARACTER	16	FCLTAB	CONVERTED LTAB
(18)	SIGNED	2	FCPSIZ	PAGE SIZE
	...1 111.		FCNUMT	"30" ..NUMBER OF TABLE ENTRIES
	...1 1.1.		FCENTL	"L'FCBNAM+L'FCLTAB+L'FCPSIZ"..LENGTH OF ENTRY
	EXPRESSION		FCTABL	"FCNUMT*FCENTL" ..LENGTH OF TABLE

How to Locate: Refer to Figure 152 on page 731 in Chapter 6, “Diagnostic Aids.”

Function Management Header 3

Definition Macro: IPW$DVD

The DSECT is used to reference the fields of a PNET TP buffer which contains a Function Management Header 3.

Bytes	Label	
Hex.	of Field	Description/Function
00	FMH3DS	Start of Function Management 3
00	FMH3LN	Length of header
01	FMH3TY	Type of header
	FMH3TYP3	X'03' - Header type 3
02	FMH3FLG	Flag byte
	FMH3FLCO	X'02' - Compaction table follows
03	FMH3MAST	Number of master characters
04	FMH3CMPT	Start of compactable characters

Generation Table (GNB) for VSE/POWER

Definition Macro: IPW$DGN

The load routine required to load IPW$$I1 and a generation table with VSE/POWER default options are supplied to the user cataloged in the system library together with all VSE/POWER phases. It is located in the initialization-processor root phase.

Should the user require other than default options, a new generation table must be assembled, and cataloged to the system library.

Offset Hex	Type	Len	Name (Dim)	Description
(0)	CHARACTER	16	GNSD	STORAGE DESCRIPTOR
(10)	CHARACTER	4	GNVR	VERSION/RELEASE/MOD LEVEL
(14)	CHARACTER	1	GNMP1	MASTER PASSWORD
(15)	BITSTRING	1	GNGFL	GENERAL FLAG BYTE
	1...		GNGQPART	"X'80'" .. Q-FILE IN PART. GETVIS
	.1..		GNGJEPU	"X'40'" .. JOBEXIT PARALLEL W-UNIT
	..1.		GNGOEPU	"X'20'" .. OUTEXIT PARALLEL W-UNIT
	...1		GNGNEPU	"X'10'" .. NETEXIT PARALLEL W-UNIT
	.... 1...		GNGXEPU	"X'08'" .. XMTEXIT PARALLEL W-UNIT
(16)	BITSTRING	2	GNACI	ACCOUNTING INFORMATION
	BITSTRING		GNACIA	"B'1000000000000000'" ..AFP ACCOUNT RECORD
	BITSTRING		GNACIC	"B'0100000000000000'" ..SAS CONNECT ACCNT RECORD
	BITSTRING		GNACIE	"B'0010000000000000'" ..EXECUTION ACCNT RECORD
	BITSTRING		GNACIL	"B'0001000000000000'" ..LIST ACCOUNT RECORD
	BITSTRING		GNACIM	"B'0000100000000000'" ..PNET XMITTER ACCNT REC.
	BITSTRING		GNACIN	"B'0000010000000000'" ..NETWORK ACCOUNT RECORD
	BITSTRING		GNACIP	"B'0000001000000000'" ..PUNCH ACCOUNT RECORD
	BITSTRING		GNACIR	"B'0000000100000000'" ..READER ACCOUNT RECORD
	1...		GNACIS	"B'0000000010000000'" ..RJE SNA ACCNT RECORD
	.1..		GNACIT	"B'0000000001000000'" ..RJE BSC ACCNT RECORD
	..1.		GNACIV	"B'0000000000100000'" ..PNET RECEIVER ACCNT REC.
	...1		GNACIX	"B'0000000000010000'" ..SAS OPERATION ACCNT REC.
	.... 1...		GNACIY	"B'0000000000001000'" ..RED. SAS OPER. ACCNT REC
	.... .1..		GNACI1	"B'0000000000000100'" ..RESERVED FOR FUTURE
	.... ..1.		GNACI2	"B'0000000000000010'" ..RESERVED FOR FUTURE
	1		GNACI3	"B'0000000000000001'" ..RESERVED FOR FUTURE
(18)	CHARACTER	8	GNSECNID	SECURITY NODEID
(20)	CHARACTER	1	GNSECAC	Security Access Control Mode
(21)	CHARACTER	23		UNUSED
(38)	CHARACTER	1		UNUSED
(39)	CHARACTER	1	GNMP2	MASTER PASSWORD
(3A)	CHARACTER	1	GNMP3	MASTER PASSWORD
(3B)	CHARACTER	1	GNMP4	MASTER PASSWORD
(3C)	CHARACTER	4	GNPTEL	XMTEXIT WORK AREA LENGTH
(40)	CHARACTER	8	GNPT	NAME OF XMTEXIT
(48)	CHARACTER	8	GNOE	OUTPUT EXIT PHASE NAME

Offset Hex	Type	Len	Name (Dim)	Description
(50)	CHARACTER	4	GNOEL	OUTPUT EXIT WORK AREA SIZE
(54)	CHARACTER	4	GNREL	READER EXIT WORK AREA SIZE
(58)	CHARACTER	4	GNPEL	PNET EXIT WORK AREA SIZE
(5C)	CHARACTER	4	GNJECL	ALTERNATE JECL PREFIX
(60)	CHARACTER	4	GNDB	DBLK VALUE
(64)	CHARACTER	2	GNDBNO	DBLK GROUP VALUE
(66)	CHARACTER	2	GNTL	TABEL LENGTH(+BSC)
(68)	CHARACTER	1	GNSL	SUBLIB
(69)	CHARACTER	1	GNJA	ACCOUNT SWITCH This byte contains a single alphabetic character; the character 'A' indicates that VSE/POWER job accounting is required; a blank character indicates that VSE/POWER accounting is not required.
(6A)	CHARACTER	1	GNPP	PAUSE PUNCH SWITCH
(6B)	CHARACTER	1	GNLG	LOG OPTION This byte contains a single alphabetic character; the character 'L' indicates that the JECL job statement is to be logged; a blank character indicates the opposite.
(6C)	CHARACTER	1	GNPY	DEFAULT PRI
(6D)	CHARACTER	1	GNNL	NUMBER OF BSC LINES
(6E)	CHARACTER	1	GNNR	NUMBER OF BSC REMOTES
(6F)	CHARACTER	1	GNSP	SPOOL MGMT SPECIFICATION
(70)	BITSTRING	1	GNOP	OPTION BYTE
	1...		GNCP	"X'80'" .. CLEAR PRINT AT EOJ
	.1..		GNMF	"X'40'" .. MARK FORM FOR SEP PAGES
	..1.		GNNS	"X'20'" .. NO SEP PAGES BTWN COPIES
				X'10' (USED IN QUEUE RECORD)
				X'08' .. RESERVED FOR FUTURE USE
				X'04' .. RESERVED FOR FUTURE USE
	.... ..1.		GNCH	"X'02'" .. CHANNEL 12
	1		GNFD	"X'01'" .. FEED FOR 3540
(71)	CHARACTER	1	GNTS	BSC TRACE AREA SIZE IN KB
(72)	CHARACTER	1	GNMP5	MASTER PASSWORD
(73)	CHARACTER	1	GNSPLIM	SPLIM VALUE
(74)	CHARACTER	8	GNTI	TIMER INTERVALS
(7C)	CHARACTER	1	GNSO	SHARED SPOOLING OPTIONS
(7D)	CHARACTER	1	GNSY	SYSTEM-ID
(7E)	CHARACTER	1	GNMP6	MASTER PASSWORD
(7F)	CHARACTER	1	GNMP7	MASTER PASSWORD
(80)	CHARACTER	16	GNLV (0)	MASTER LIST VALUES
(80)	CHARACTER	1	GN#PERF	NO. PERFORATION LINES FOR $$I2 PROCESS SET **LINE=N
(81)	CHARACTER	1		RESERVED

Offset Hex	Type	Len	Name (Dim)	Description
(82)	CHARACTER	1	GNFLG	FLAG BYTE EQUATES DEFINED IN THE DMB
(83)	CHARACTER	1	GNJL	JSEP LIST
(84)	CHARACTER	4	GNRL	RBS LIST
(88)	CHARACTER	4	GNL1	STDLINE FIRST
(8C)	CHARACTER	4	GNL2	STDLINE SECOND
(90)	CHARACTER	16	GNPV (0)	MASTER PUNCH VALUES
(90)	CHARACTER	3		RESERVED
(93)	CHARACTER	1	GNJP	JSEP PUNCH
(94)	CHARACTER	4	GNRP	RBS PUNCH
(98)	CHARACTER	4	GNC1	STDCARD FIRST
(9C)	CHARACTER	4	GNC2	STDCARD SECOND
(A0)	CHARACTER	8	GNRE	RDREXIT NAME
(A8)	SIGNED	2	GN#M	MAX NO MSG IN NTFY-QUEUE
(AA)	CHARACTER	8	GNNT	NETWORK TABLE NAME
(B2)	CHARACTER	8	GNPR	PNET USER RDR-EXIT
(BA)	CHARACTER	1	GNRJ	RJEBSC SPECIFICATION
(BB)	CHARACTER	8	GNMT	MEMBER TYPE SPECIFICATION
(C3)	CHARACTER	1	GNMP8	MASTER PASSWORD
WORK AREA FOR EXTRACT MACRO Referred to by label GNWE. The EXTRACT macro (SVC98) will be issued from the VSE/POWER load routine which loads IPW$$IP. Information about the VSE/POWER partition will be saved in following fields.				
(C4)	CHARACTER	20	GNWE (0)	
(C4)	SIGNED	4	GNPB	PARTITION BEGIN ADDRESS
(C8)	SIGNED	4	GNPE	VIRT. PART. END GETVIS EXCL
(CC)	SIGNED	4	GNPG	VIRT PART END GETVIS INCLUD.
(D0)	SIGNED	4	GNFX	PFIX LIMIT IN K-BYTES
(D4)	SIGNED	4	GNFC	PFIX COUNT IN NR OF PAGES
(D8)	SIGNED	4	GNGB	GETVIS AREA BEGIN ADDRESS
(DC)	SIGNED	4	GNGE	GETVIS AREA END ADDRESS
(E0)	CHARACTER	8	GNIN	'IPW$$IP' PHASE NAME
During initialization the IPW$$IP phase name is overlaid by the following fields:				
(E0)	SIGNED	4	GNRM	ADDR REMOTE CONTROL BLOCK
(E4)	SIGNED	4	GNSS	ADDR SUCB SPACE
(E8)	CHARACTER	13	GNLT	LTAB
(F5)	ADDRESS	3	GNLU	LENGTH OF LU TABLE
SNA Information				
(F8)	CHARACTER	32		SNA INFORMATION
(F8)	CHARACTER	2	GNTT	TOTAL TBL LENGTH(+SNA+BSC)
(FA)	CHARACTER	1		RESERVED

Offset Hex	Type	Len	Name (Dim)	Description
(FB)	CHARACTER	1	GNAL	LENGTH ACB PASSWORD
(FC)	CHARACTER	8	GNAP	ACB PASSWORD
(104)	CHARACTER	1	GNSU	MAX NR OF SNA WORKSTATIONS GNSU will be overlaid by GNSR during initialization if the number of SNA remotes is smaller than the maximum number of SNA logical units.
(105)	CHARACTER	1	GNSR	NR OF SNA REMOTES
(106)	CHARACTER	1	GNFR	FIRST SNA REMOTE ID
(107)	CHARACTER	1	GNHR	LAST SNA REMOTE ID
(108)	ADDRESS	1	GNVA	LENGTH,APPLID FOR VTAM
(111)	CHARACTER	3		RESERVED
(114)	ADDRESS	4	GNLA	ADDR OF FIRST LINE BLOCK
	EXPRESSION		GNEN	"*" END OF TABLE
	EXPRESSION		GNLN	"GNEN-GNDS" LENGTH OF TABLE ASM H V 02 15.31

Initialization Processor Work Area (IP)

Definition Macro: IPW$DEF IWK=MAP

This area contains addresses and information that are used in communication among the initialization root phase and various initialization phases. It is located in the initialization-processor work phase (IPW$$IP).

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
				INITIALIZATION PROCESSOR WORKAREA	
(0)	0	STRUCTURE	0	IWKDS	, IP WORKAREA LAYOUT
(0)	0	SIGNED	4	IWKRSA (12)	REGISTER SAVE AREA
(30)	48	ADDRESS	4	IWIPSA	START ADDRESS OF ROOTPHASE
(34)	52	ADDRESS	4	IWIPEA	END ADDR OF ROOTPHASE
(38)	56	ADDRESS	4	IWPPEA	END ADDR OF LAST PHASE
(3C)	60	ADDRESS	4	IWMAXA	MAX ADDRESS FOR IPLOAD
(40)	64	ADDRESS	4	IWRDRS	JOB EXIT SIZE
(44)	68	ADDRESS	4	IWOUTS	OUT EXIT SIZE
(48)	72	ADDRESS	2	IWLALN	LENGTH OF OVERLAY AREA
(4A)	74	CHARACTER	80	IWMSGAR	MESSAGE I/O AREA
		.1.. .111		IWMSGLN	"71" .. MSG LENGTH (SHORT FORM)
(9A)	154	CHARACTER	80	IWCDIN	CARD READ IN AREA
(EA)	234	ADDRESS	1	IWABSW	ABNORMAL WARMSTART SWITCH
		.1..		IWABOK	"C' '" .. NO RECOVERY NEEDED
		11.. ...1		IWABST	"C'A'" .. FULL RECOVERY NEEDED
		11.1 .111		IWABPA	"C'P'" .. PARTIAL RECOVERY NEEDED
(EB)	235	BITSTRING	1	IWFLG1	IP FLAG BYTE 1
		1...		IWOSUP	"X'80'" .. OTHER SYSTEM UP IND
		1		IWREQT	"X'01'" .. REQUEST TERMINATION
		.... ..1.		IWSIPH	"X'02'" .. SINGLE LOAD AFFECTED
		.... .1..		IWCANCEL	"X'04'" .. LOAD ROUTINE CANCELLED
		.... 1...		IWSHORT	"X'08'" .. AREA TO SMALL (IPLOAD)
		...1		IWFEAT	"X'10'" .. FEATURE TO BE LOADED
		..1.		IWNORUN	"X'20'" .. DISP=X FOR EXECUT. Q-SET
		.1..		IWNOTIT	"X'40'" .. NO TIMER TASK EXISTS
(EC)	236	BITSTRING	1	IWFLG2	IP FLAG BYTE 2
		1...		IWIOMR	"X'80'" .. MASTER RECORD I/O ERROR
		.1..		IWIOQRB	"X'40'" .. I/O ERROR QUEUE REC BLOCK
		..1.		IWEXIT	"X'20'" .. JOBEXIT PROCESSING ACTIVE
		...1		IWPSVA	"X'10'" .. PHASE IS LOADED IN SVA
		.... 1...		IWF2NSW	"X'08'" .. SWITCH TO NON SHARED SYS
(ED)	237	BITSTRING	1	IWJCMQ	JCM QUEUE SIZE
(EE)	238	BITSTRING	2	IWMSP	MISSING SPACE SAVE FIELD
(F0)	240	ADDRESS	4	IWGENA	ADDRESS ORIGINAL GEN TABLE
	DTF ADDRESS TABLE PART #1				
(F4)	244	ADDRESS	4	IWQFILE	ADDRESS OF QUEUE FILE DTF
(F8)	248	ADDRESS	4	IWDFILE	ADDRESS OF DATA FILE DTF
(FC)	252	ADDRESS	4	IWAFILE	ADDRESS OF ACCOUNT FILE DTF
(100)	256	ADDRESS	4	IWINPTF	ADDRESS OF SYSIPT DTF
(104)	260	ADDRESS	4	IWPWRLK	ADDR OF QUEUE FILE DTF CB
	SUBROUTINE ADDRESS TABLE PART #1				
(108)	264	ADDRESS	4	IWIDAL	ADDR OF IDAL BUILD ROUTINE
(10C)	268	ADDRESS	4	IWSUPH	ADDR SET UP DTFPH ROUTINE
(110)	272	ADDRESS	4	IWPRST	ADDR PRINT STATUS REPORT RTN
(114)	276	ADDRESS	4	IWFMCB	ADDR FORMAT MCB RTN
(118)	280	ADDRESS	4	IWCOLD	ADDR OF COMMON LOAD
(11C)	284	ADDRESS	4	IWCMSG	ADDR OF COMMON MSG ROUTINE
(120)	288	ADDRESS	4	IWGENT	ADDRESS VSE/POWER GEN TABLE

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(124)	292	ADDRESS	4	IWMCBC	ADDR OF SKELETON MCB (C-K-D)
(128)	296	ADDRESS	4	IWMCBF	ADDR OF SKELETON MCB (FBA)
TEMPORARY TRACE ADDRESS TABLE					
(12C)	300	ADDRESS	4	IWTBA	TRACE BEGIN ADDRESS (TEMP)
(130)	304	ADDRESS	4	IWTEA	TRACE END ADDRESS (TEMP)
ADDRESS OF EBCDIC/ASCII TABLES					
(134)	308	ADDRESS	4	IWEBAS	EBCDIC/ASCII TABLE BEGIN
ADDRESS OF USER EXITS					
(138)	312	ADDRESS	4	IWRXEP	RDR EXIT ENTRY POINT ADDR
(13C)	316	ADDRESS	4	IWOXEP	OUT EXIT ENTRY POINT ADDR
(140)	320	ADDRESS	4	IWNXEP	NET EXIT ENTRY POINT ADDR
(144)	324	ADDRESS	4	IWXXEP	XMT EXIT ENTRY POINT ADDR
(148)	328	ADDRESS	4	IWRXLP	RDR EXIT LOAD POINT ADDR
(14C)	332	ADDRESS	4	IWOXLP	OUT EXIT LOAD POINT ADDR
(150)	336	ADDRESS	4	IWNXLP	NET EXIT LOAD POINT ADDR
(154)	340	ADDRESS	4	IWXXLP	XMT EXIT LOAD POINT ADDR
MISCELLANEOUS FIELDS					
(158)	344	BITSTRING	1	IWJMQM	JOB EVENT QUEUE MULTIPLIER
(159)	345	BITSTRING	3		FREE FOR FUTURE USE
(15C)	348	BITSTRING	4	IWMRT	MESSAGE ROUTING CODE
(160)	352	BITSTRING	2	IWMDC	MESSAGE DESCRIPTOR CODE
(162)	354	BITSTRING	8	IWCPUID	CPUID OBTAINED BY EXTRACT
SUBROUTINE ADDRESS TABLE PART #2					
(16C)	364	ADDRESS	4	IWFD00	ADDR FORMAT DATA FILE RTN
(170)	368	ADDRESS	4		RESERVED
(174)	372	ADDRESS	4		RESERVED
(178)	376	ADDRESS	4		RESERVED
DTF ADDRESS TABLE PART #2					
(17C)	380	ADDRESS	4	IWDTEST	ADDRESS OF TEST FILE DTF
(180)	384	ADDRESS	4	IWQFOLD	ADDRESS OF OLD Q-FILE DTF
(184)	388	ADDRESS	4	IWQTEST	ADDRESS OF TEST Q-FILE DTF
(188)	392	ADDRESS	4	IWPWRL2	ADDRESS DTF OF OLD Q-FILE
EQUATE STATEMENTS THE FOLLOWING EQUATE STATEMENTS PROVIDE NECESSARY RESOLUTION FOR UNDEFINED SYMBOLS WITHIN THE DUMMY SECTION DEFINITION.					
		....		IPCS	"0" START ADDRESS OF IP MAIN RTN
		....		IPEND	"0" END ADDRESS OF IP ROOTPHASE
		....		IPLDARLN	"0" LENGTH OF OVERLAY LOAD AREA
		....		IJQFILE	"0" QUEUE FILE DTF
		....		IJDFILE	"0" DATA FILE DTF
		....		IJAFILE	"0" ACCOUNT FILE DTF
		....		IJSYSIN	"0" SYSIPT DTF
		....		IPDTLQF	"0" QUEUE FILE DTL
		....		IDA00	"0" BUILD IDAL ROUTINE
		....		SD00	"0" SET UP DTFPH ROUTINE
		....		PS00	"0" PRINT STATUS REPORT RTN
		....		FM00	"0" FORMAT MCB QUEUE/DATA FILE
		....		IPLOAD	"0" COMMON LOAD ROUTINE
		....		IPLMSG	"0" COMMON MESSAGE ROUTINE
		....		FD00	"0" FORMAT DATA FILE
		....		IJDTEST	"0" TEST FILE DTF
		....		IJQFOLD	"0" OLD QFILE DTF
		....		IJQTEST	"0" TEST QFILE DTF
		....		IJDTLQ2	"0" 2ND QFILE LOCK DTL
		....		GNCB	"0" ADDR VSE/POWER GEN TABLE
		....		D1SD	"0" C-K-D MCB SKELETON
		....		F1SD	"0" FBA MCB SKELETON
		....		EBAS	"0" EBCDIC-ASCII TABLE

Journal Communication Area (JCA)

Definition Macro: IPW$DEF JCA=YES

This area is used to communicate between the POFFLOAD BACKUP|SAVE|PICKUP task and the Journaling task (running as a Print Status task).

The POFFLOAD task messages are written to the JCA which are then read by the Journaling task and then written to the $OFJnnnn LST entry. The JCA also has trace information for error determination and task status indicators.

```
====================================================================================
                   JOURNAL COMMUNICATION AREA (JCA)
====================================================================================

  OFFSET   OFFSET
 DECIMAL      HEX  TYPE          LENGTH  NAME (DIM)     DESCRIPTION
======== ======== ========= ========  ============== ==============================
       0      (0) STRUCTURE        0  JCADS          POFFLOAD JOURNALING
                                                       COMMUNICATION AREA
       0      (0) CHARACTER        8   JCAEYE        EYECATCHER = 'JCA'
       8      (8) BITSTRING        2   JCATROF       $$OF TRACE BYTE STATE/REQ (OF)
      10      (A) BITSTRING        2   JCATROFS      $$OF SUBR TRACE BYTE STATE/REQ
                                                     (OFS)
      12      (C) BITSTRING        6   JCATRPS       $$PS TRACE BYTE STATE/REQ
                                                     (OF+OFS+PS)
      18     (12) BITSTRING        8   JCATRTRO      $$TR TRACE BYTE:$$OF
                                                     CALL(OF+OFS+PS+TR)
      26     (1A) BITSTRING        8   JCATRTRP      $$TR TRACE BYTE:$$PS
                                                     CALL(OF+OFS+PS+TR)
      34     (22) BITSTRING        2   JCATROUT      $$PS TIMEOUT TRACE BYTES (FOR
                                                     1Q5MI)
      36     (24) BITSTRING        1   JCAFLG        FLAG BYTE
                  1... ....            JCAF1Q2A      "X'80'" -$$TR TO ISSUE JOURNAL
                                                     VERSION OF 1Q2AI
                  .1.. ....            JCAF1Q5L      "X'40'" -$$TR TO ISSUE JOURNAL
                                                     VERSION OF 1Q5LI
                  ..1. ....            JCAFMSG       "X'20'" -$$OF TO ISSUE JOURNAL
                                                     MSG IN JCATEXT
                  ...1 ....            JCAFIOPS      "X'10'" -$$TR CALLED FOR
                                                     Q/DFILE I/O FOR $$PS
                  .... 1...            JCAFIOOF      "X'08'" -$$TR CALLED FOR
                                                     Q/DFILE I/O FOR $$OF18
                  .... .1..            JCAFNOUP      "X'04'" -$$OF IND FOR $$PS-NO
                                                     INCRE ENTRY NUM36
      37     (25) CHARACTER        1   JCAREQPS      $$OF JOURNALING REQUEST TO $$PS
                  11.. 1..1            JCAREQI       "C'I'" -'I'=INITIATE
                  11.1 .11.            JCAREQP       "C'O'" -'P'=WRITE REST OF
                                                     PROLOG
                  11.. .11.            JCAREQF       "C'F'" -'F'=FORMAT AND WRITE
                                                     QREC,
                  111. .1.1            JCAREQV       "C'V'" -'V'=WRITE BEGIN NEW
                                                     VOLUME LINE
                  111. .111            JCAREQX       "C'X'" -'X'=WRITE TEXT LINE IN
                                                     JCATEXT
                  11.. ..11            JCAREQC       "C'C'" -'C'=CLOSE-WRITE EPILOG
                                                     LINES EOJ
                  111. ..11            JCAREQT       "C'T'" -'T'=TERMINATE (IN
                                                     IPW$$TR)
      38     (26) CHARACTER        1   JCARUNAO      ACTUAL RUNNING STATE $$OF
                                                     JOURNALING
                  11.. 1...            JCARUNOH      "C'H'" -'H'=ATTACHING, TASK TO
                                                     BE ATTACHED
                  11.. 1..1            JCARUNOI      "C'I'" -'I'=INITIALIZING,TASK
                                                     ATTACHED
                  11.1 1..1            JCARUNOR      "C'R'" -'R'=RUNNING $$PS
                                                     RUNNING
```

		11.. ..11		JCARUNOC	"C'C'" -'C'=CLOSED
		11.. .1.1		JCARUNOE	"C'E'" -'E'=CLOSE_ERR ($$PS ERR+$$PS TERM OK)
		11.1 .111		JCARUNOP	"C'P'" -'P'=CLOSE_STOP'G (CLOSE+$$PS
		111. ..1.		JCARUNOS	"C'S'" -'S'=STOPPING ($$PS NO TERM)
		11.. ...1		JCARUNOA	"C'A'" -'A'=ABEND (CRAZY SITUATION)
39	(27)	CHARACTER	1	JCARUNAP	ACTUAL RUNNING STATE $$PS
		11.. 1...		JCARUNPH	"C'H'" -'H'=ATTACHING, TASK TO BE ATTACHED
		11.. 1..1		JCARUNPI	"C'I'" -'I'=INITIALIZING,TASK ATTACHED
		11.1 1..1		JCARUNPR	"C'R'" -'R'=RUNNING TASK RUNNING
		11.. ..11		JCARUNPC	"C'C'" -'C'=CLOSED LST CLOSED
		11.. .111		JCARUNPG	"C'G'" -'G'=TERMINATING TASK TERMINATING
		111. ..11		JCARUNPT	"C'T'" -'T'=TERMINATED TASK TERMINATED
		11.. ..1.		JCARUNPA	"C'B'" -'A'=ABENDING TASK INTERNAL ERROR
40	(28)	BITSTRING	2	JCAPSRC	TERMINATION RETURN CODE $$PS
42	(2A)	BITSTRING	1	JCARUNPS	RUNNING STATE $$PS SAVED AT $$TR BGN 23
43	(2B)	CHARACTER	5	JCAJOBNO	JOURNALING OUTPUT JOBNUMBER
48	(30)	CHARACTER	8	JCAJOBNM	JOURNALING OUTPUT JOBNAME '$OFJNNNN'
56	(38)	ADDRESS	4	JCATCBOP	$$OF TASK TCB ADDR
60	(3C)	ADDRESS	4	JCATCBPP	$$PS TASK TCB ADDR
64	(40)	ADDRESS	4	JCATEMP	TEMP WORKAREA
68	(44)	CHARACTER	16	JCATCBOD	$$OF TCB DESCRIPTOR
84	(54)	CHARACTER	16	JCATCBPD	$$PS TCB DESCRIPTOR
100	(64)	ADDRESS	4	JCAECBOF	$$OF ECB
104	(68)	ADDRESS	4	JCAECBPS	$$PS ECB
108	(6C)	ADDRESS	4	JCAECBTQ	$$OF TIMER INTERRUPT ECB
112	(70)	SIGNED	2	JCAECBL(0)	MULTIPLE-WAIT LIST FOR IPW$WFM MACRO
112	(70)	ADDRESS	4	JCAECBLT	-TIMER ELEMENT ECB
116	(74)	ADDRESS	4	JCAECBLO	-$$OF ECB: JCAECBOF
120	(78)	ADDRESS	4	JCAECBLE	X'FF' - END OF MULTIPLE-WAIT LIST
124	(7C)	CHARACTER	8	JCABGDA	BEGIN DATE 'MM/DD/YY'(CREATING SYS FMT)
132	(84)		4	JCABGTI	BEGIN TIME PACKED (0HHMMSSF)
136	(88)	ADDRESS	4	JCAITEMN	RUNNING JOURNAL ITEM =NNNNNN
140	(8C)	BITSTRING	2	JCAVOLN	RUNNING VOL. COUNT VOL=NNNNN
142	(8E)	CHARACTER	1	JCAQID	QUEUE ID BEING DISPLAYED 32
143	(8F)	BITSTRING	1	JCATEXTL	JOURNALING TEXT MSG LENGTH (SET BY IPW$GAM DEST=JCATEXTL OR $$PS)
144	(90)	CHARACTER	132	JCATEXT	JOURNAL DISPLAY TEXT 25 (132 BYTES FOR $GAM SUB=YES) 25

```
     276    (114) CHARACTER       80   JCAVOL1       LABELED TAPE VOL1
     356    (164) CHARACTER       80   JCAHDR1       LABELED TAPE HDR1
     436    (1B4) CHARACTER      130   JCACMD        *POFFLOAD OPERATOR COMMAND
     566    (236) CHARACTER        8   JCACMDT       *POFFLOAD TYPE
     574    (23E) CHARACTER      132   JCA1Q4C       COPY OF MSG 1Q4CI 'DATE BEGIN
                                                     ...'
     708    (2C4) ADDRESS          4   *(0)
     708    (2C4) BITSTRING       24   JCATQEO       TQE ELEMENT
     732    (2DC) ADDRESS          4   *(0)
     732    (2DC) BITSTRING      624   *             UNUSED 27
    1356    (54C) ADDRESS          4   JCATRRD       IPW$$TR REG. R13 FOR JOURSUB
    1360    (550) ADDRESS          4   JCAOFRD       IPW$$OF REG. R13 FOR JOURSUB
    1364    (554) ADDRESS          4   *(12)         UNUSED 27
    1412    (584) ADDRESS          4   JCAREGS2(12)  REGISTER SAVE AREA 2 $$OF
    1460    (5B4) ADDRESS          4   JCAREGSM(12)  REGISTER SAVE AREA 3 $$OF $GAM
    1508    (5E4) ADDRESS          4   *(12)         UNUSED 27
    1556    (614) ADDRESS          4   *(12)         UNUSED 27
    1604    (644) ADDRESS          4   JCAREGT2(12)  REGISTER SAVE AREA 2 $$TR
    1652    (674) ADDRESS          4   JCAREGTM(12)  REGISTER SAVE AREA 3 $$TR $GAM
    1700    (6A4) ADDRESS          4   JCAREGTW(12)  REGISTER SAVE AREA 3 $$TR $WTO
    1748    (6D4) ADDRESS          4   *(12)         UNUSED 27
    1796    (704) ADDRESS          4   *(12)         UNUSED 27
    1844    (734) ADDRESS          4   JCAREGTF(0)   REGISTER SAVE AREA FOR FUNCTION
                                                      CALL
    1900    (76C) CHARACTER        4   JCATCCU       POFFLOAD TCB TCUU
    1904    (770) ADDRESS          4   JCAWAITX      $$PS WAIT MAX (FROM OFJOUSTM)
    1904    (770)                      JCADSLEN      "*-JCADS" LENGTH OF INTERFACE WA
```

Logical Data Record Area (LDA)

Definition Macro: IPW$DDR

This area is used to hold data which is to be written to the data file (write operation) and read from the data file (read operation). Its size is set by the DBLK parameter.

Records are transferred to the LDA one at a time from the PDA for read and for write operations. When the LDA is full, or there is no more room for a complete record, the information is written to or read from the data file. It is addressed via the I/O request word in the TCB, and each record is addressed via the channel program in the MCB for the data file.

The format of a logical data record is as follows.

```
Bytes       Label
Hex.        of Field    Description/Function
---------------------------------------------------------------------
            DRDS        Definition of this dummy section
00-01       DRRL        Logical record length.
                        This field contains the length of the
                        data-record text with preceding fields like
                        DRRL, DRGP, DRGP2, DRGP3, and DRCC.
02          DRGP        General purpose byte:
                        X'00' normal record
                        X'01' line print/card move data
                        X'02' 3540 data record
                        X'04' end of data
                        X'08' reserved  - must not be used
                        X'10' end of block
                        X'20' end of 3540 data
                        X'40' extended record
03          DRCC        Command code.
                        Indicates command code for output
                        list/punch device or 00 when input
                        record or spooled-account record.
                        Special types:
                          X'FF' .. CONTROL RECORD
                          X'FE' .. NEW FORMS
04          DRG2        General purpose byte 2
                        X'80' job header record
                        X'40' job trailer record
                        X'20' data set header record
                        X'10' CPDS data record
                        X'08' Unused
                        X'04' Fixed format message record
                        X'02' ASA  data record
05          DRG3        General purpose byte 3
                        X'80' extended record begin
                        X'40' extended record middle
                        X'20' extended record end
06-07       DREL        Extended record residual length not including DRDL
            DRDL        Length of descriptor = addr(* - DRRL)
            DRDT        Text of data record
```

How to Locate: Refer to Figure 152 on page 731 in Chapter 6, “Diagnostic Aids.”

Logical Reader Work Area

Definition Macro: IPW$DLW

The logical reader work area is used by IPW$$SC, IPW$$LR and IPW$$NR during checking of the time event scheduling parameters of the * $$ JOB statement.

The format of a logical reader work area is as follows.

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
LOGICAL READER WORK AREA					
(0)	0	CHARACTER	144	LRDS1 (0)	1 ST SAVE AREA
(0)	0	CHARACTER	15	LRDD (0)	START OF INFO
(0)	0	BITSTRING	1	LRDDGP1	GENERAL PURPOSE BYTE 1
		1...		LRDDGP1R	"X'80'" RERUN=NO SPECIFIED
		.1..		LRDDGP1R	X'40' RESERVED
		..1.		LRDDGP1R	EQU X'20' RESERVED
		...1		LRDDGP1F	"X'10'" DUEFRQ SPECIFIED
		.... 1...		LRDDGP1T	"X'08'" DAILY SPECIFIED
		.... .1..		LRDDGP1W	"X'04'" WEEKDAYS SPECIFIED
		.... ..1.		LRDDGP1D	"X'02'" DAYS WITHIN MONTH
		1		LRDDGP1M	"X'01'" MONTH WITHIN YEAR
(1)	1	BITSTRING	1	LRDDGP2	GENERAL PURPOSE BYTE 2 (EQUATES DEFINED IN QUEUE RECORD)
(2)	2	BITSTRING	2	LRDDMY	MONTHS WITHIN YEAR
(4)	4	BITSTRING	4	LRDDDM	DAYS WITHIN MONTH
(8)	8	CHARACTER	4	LRDDN (0)	START OF NEXT DUE DATE PACKED DECIMAL WITHOUT SIGN
(8)	8	BITSTRING	2	LRDDNY	YEAR (1988-2087)
(A)	10	BITSTRING	1	LRDDNM	MONTH (1-12)
(B)	11	BITSTRING	1	LRDDND	DAY (1-31)
(C)	12	BITSTRING	2	LRDDNT	NEXT DUE TIME
(E)	14	BITSTRING	1	LRDDFQM	MINUTES (0-59) OF DUEFRQ
		 1111		LRSEC	"*"
(4)	4	BITSTRING	1	LRDDDW	DAYS WITHIN A WEEK
(2)	2	CHARACTER	2	LRDDFT (0)	START OF FIRST TIME
(2)	2	BITSTRING	1	LRDDFH	HOUR (0-23)
(3)	3	BITSTRING	1	LRDDFM	MINUTE (0-59)
(4)	4	BITSTRING	1		USED FOR WEEKDAYS
(5)	5	CHARACTER	2	LRDDLT (0)	START OF LAST TIME
(5)	5	BITSTRING	1	LRDDLH	HOUR (0-23)
(6)	6	BITSTRING	1	LRDDLM	MINUTE (0-59)
(7)	7	BITSTRING	1	LRDDFQH	HOURS (0-23) OF DUEFRQ (EQUATES DEFINED IN QUEUE RECORD)
ADDITIONAL WORK FIELDS					
(F)	15	BITSTRING	1	LRFLG1	FLAG BYTE 1
		1...		LRF1DDM	"X'80'" .. DDMMYY FORMAT
		.1..		LRF1DAS	"X'40'" .. RANGE VALUE SPECIFIED
		..1.		LRF1NU	"X'20'" .. NUMERIC OPERAND
		...1		LRF1FTSW	"X'10'" .. FIRST TIME SWITCH
(10)	16	BITSTRING	1	LRFLG2	FLAG BYTE2
		1...		LRF2DISP	"X'80'" .. DISPOSITION SPECIFIED
		.1..		LRF2D	"X'40'" .. DUE TIME PRESENT
		..1.		LRF2DD1	"X'20'" .. DUE DATE PRESENT
		...1		LRF2R	"X'10'" .. RERUN SPECIFIED
		.... 1...		LRF2TESP	"X'08'" .. TIME EVENT SCHEDULING PRESENT
		.... .1..		LRF2DIV	"X'04'" .. DUE INTERVAL PRESENT

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(14)	20	SIGNED	4		RESERVED FOR FUTURE USE
(18)	24	SIGNED	4	LRSVRE	SAVE AREA FOR RETURN REGISTER
(1C)	28	SIGNED	2	LRSTRT	WORK FIELD FOR COMBINED FORMAT
(1E)	30	SIGNED	2	LRSTOP	WORK FIELD FOR COMBINED FORMAT
(20)	32	SIGNED	4	LRW1	FOR PACK PURPOSES
(24)	36	SIGNED	4	LRW2	FOR PACK PURPOSES
(28)	40	SIGNED	4	LRW3	WORK FIELD
(2C)	44	BITSTRING	1	LROFF	SAVE OFFSET OF BIT SETTING
(2D)	45	BITSTRING	3		RESERVED FOR FUTURE USE
(30)	48	SIGNED	4	LROFFMD	OFFSET INTO MONTH/DAY TABLE
(34)	52	SIGNED	4	LRSY	SAVE SPECIFIED YEAR
(38)	56	SIGNED	4	LRSDADDR	AREA TO SAVE DELIM ADDRESS
(3C)	60	SIGNED	4	LRWAMSK	AREA TO SAVE BIT MASK
OPERAND AREA USED BY FORMAT ROUTINE IN $SC					
(40)	64	CHAR-ACTER	34	LROP1 (0)	OPERAND VALUE 1
(40)	64	BITSTRING	1	LROP1LEN	LENGTH OF OPERAND CONTENTS
(41)	65	BITSTRING	1	LROP1SW	FLAG BYTE
(42)	66	BITSTRING	1		FLAG BYTE 2
(43)	67	BITSTRING	1		MASK BYTE
(44)	68	CHAR-ACTER	24		OPERAND VALUE
(5C)	92	BITSTRING	2	LROP1HEX	HEXADECIMAL VALUE OF OPERAND
(5E)	94	BITSTRING	4	LROP1DEC	DECIMAL VALUE OF OPERAND
(62)	98	CHAR-ACTER	34	LROP2 (0)	OPERAND VALUE 2
(62)	98	BITSTRING	1	LROP2LEN	LENGTH OF OPERAND CONTENTS
(63)	99	BITSTRING	1	LROP2SW	FLAG BYTE
(64)	100	BITSTRING	1		FLAG BYTE 2
(65)	101	BITSTRING	1		MASK BYTE
(66)	102	CHAR-ACTER	24		OPERAND VALUE
(7E)	126	BITSTRING	2	LROP2HEX	HEXADECIMAL VALUE OF OPERAND
(80)	128	BITSTRING	4	LROP2DEC	DECIMAL VALUE OF OPERAND
		1... .1..		LRWALN1	"*-LRDS" LENGTH OF WORK AREA TO CLEAR
(84)	132	SIGNED	4	LRRCM	RETURN MESSAGE
(88)	136	SIGNED	4	LRRCC	RETURN ERROR CODE
(8C)	140	SIGNED	4	LRNRBREG	SAVE BASE REG FOR $$NR
(90)	144	CHAR-ACTER	4	LRDS2 (0)	2 ND SAVE AREA
(90)	144	SIGNED	4	LRSAVRD	SAVE AREA FOR R13 IN $$LR
		1..1 .1..		LRWALN	"*-LRDS" LENGTH OF WORK AREA

Logical Writer Work Space

Definition Macro: IPW$DEF ACCT=YES

The logical writer work space contains counters used for accounting purposes and control fields for the logical writer (IPW$$LW). The work area is acquired by the logical writer routine at the beginning of a job and released at end-of-job processing.

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
					LOGICAL WRITER WORK AREA
(0)	0	STRUCTURE	0	LADS	
	COUNTERS FOR ACCOUNTING PURPOSES				
(0)	0	SIGNED	4	LATL	TOTAL LINE OR CARD COUNT (R7)
(4)	4	SIGNED	4	LAEL	EXTRA LINES/CARDS
(8)	8	SIGNED	4	LACL	CURRENT LINE/CARD
(C)	12	SIGNED	4	LARL	RESTART CURRENT PAGE/CARD
(10)	16	SIGNED	4	LAEP	EXTRA PAGES
(14)	20	SIGNED	4	LATP	TOTAL PAGES FROM DATA FILE
(18)	24	SIGNED	4	LACP	CURRENT PAGE
(1C)	28	SIGNED	4	LARP	RESTART PAGE COUNT
(20)	32	SIGNED	4	LATR#	TOTAL RECORDS
(24)	36	SIGNED	4	LACR#	CURRENT RECORD NUMBER
(28)	40	SIGNED	4	LAER#	EXTRA RECORD NUMBER
(2C)	44	SIGNED	4	LARR#	RESTART CURRENT RECORD NUMBER
(30)	48	SIGNED	4	LAST	TASK START TIME (0HHMMSSF)
(34)	52	BITSTRING	1	LASR	START AFTER PSTOP CUU,RESTART INDIC.
(35)	53	BITSTRING	1	LAWS	WORKFIELD FOR COPY COUNT
(36)	54	BITSTRING	1	LWTRU	TYPE OF RESTART UNIT (L/P/R)
(37)	55	BITSTRING	3		UNUSED
	LOGICAL WRITER CONTROL INFORMATION				
		THE FOLLOWING FIELDS ARE USED BY THE LOGICAL WRITER. THEY ARE USED TO CONTROL PRINTING OF DATA WITHOUT MACHINE CONTROL CHARACTER. EACH DATA RECORD IS WRITTEN WITH THE 'WRITE AND SPACE' OP CODE. FURTHERMORE SOME HELP FIELDS ARE HERE DEFINED TO SAVE INFORMATION DURING PROCESSING OF A QUEUE ENTRY.			
(3A)	58	ADDRESS	1	LWILNCT	DEFAULT LINE COUNT (PAGE SIZE)
		1111 1111		LWINOCT	"X'FF'" .. CAUSES NOT TO COUNT LINES
(3B)	59	ADDRESS	1	LWICLCT	CURRENT LINE COUNT
(3C)	60	BITSTRING	1	LWIFLG1	FLAG BYTE 1
		1...		LWIF1FT	"X'80'" ..FIRST TIME SWITCH
		.1..		LWIF1IN	"X'40'" ..INSERT SKIP TO CH 1 IN PROGR
		..1.		LWIF1EOD	"X'20'" ..EOD TO PASS
		...1		LWIF1OV	"X'10'" ..PRINT OVERFLOV INDICATOR
		.... 1...		LWIF1PE	"X'08'" ..POSITION AT END OF Q-ENTRY
		.... .1..		LWIF1SPM	"X'04'" ..SPOOLING POINTERS MODIFIED
		.... ..1.		LWIF1IAR	"X'02'" ..ASA CONVERSION IN PROGRESS
		1		LWIF1NSP	"X'01'" ..NO SEPERATORS PRODUCED
(3D)	61	BITSTRING	1	LWIFLG2	FLAG BYTE 2
		1...		LWIF2EOD	"X'80'" ..EOD RECORD PASSED
		.1..		LWIF2SQE	"X'40'" ..START QUEUE ENTRY
		..1.		LWIF2SNC	"X'20'" ..START NEXT COPY
		...1		LWIF2FCB	"X'10'" ..CALCULATE LINES/PAGE
		.... 1...		LWIF2SSP	"X'08'" ..SUPPRESS SEPERATOR PAGE TEST
(3E)	62	ADDRESS	2	LWISVRL	SAVED ORIGINAL RECORD LENGTH
(40)	64	ADDRESS	4	LWSERWKA	PTR TO WORKAREA CONTAINING SER RECORD FOR PRINTER SETUP
(44)	68	BITSTRING	8	LWTCRW	AREA TO SAVE TCRW
(4C)	76	SIGNED	4	LWRE	SAVE RE

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(50)	80	SIGNED	4	LWRF	SAVE RF
	SEPARATOR PAGE CONTROL INFORMATION				
(54)	84	CHARACTER	2	LWIPART	PARTITION IDENTIFIER
(56)	86	CHARACTER	1	LWISYSID	VSE/POWER SYSTEM ID
(57)	87	BITSTRING	1		RESERVED
(58)	88	ADDRESS	4	LWIWRKA	PTR WORKAREA FOR SEPARATOR RTN
(5C)	92	CHARACTER	8	LWINACCT	NETWORK ACCOUNT NUMBER
(64)	100	CHARACTER	8	LWIEXNOD	EXECUTION NODE NAME
(6C)	108	CHARACTER	20	LWIPGRNM	PROGRAMMER NAME
(80)	128	CHARACTER	8	LWIBLDG#	BUILDING NUMBER
(88)	136	CHARACTER	8	LWIROOM#	ROOM NUMBER
(90)	144	CHARACTER	8	LWIDEPT#	DEPARTMENT NUMBER
(98)	152	CHARACTER	8	LWIFCBNM	FCBNAME FOR SAS
(A0)	160	SIGNED	2		UNUSED
	OUTPUT EXIT WORK AREA				
(A2)	162	BITSTRING	16	LWOTCRW	AREA TO SAVE TCRW
(B4)	180	SIGNED	4	LWOINSR	NUMBER OF INSERTED RECORDS
(B8)	184	SIGNED	4	LWODELR	NUMBER OF DELETED RECORDS
(BC)	188	SIGNED	4	LWOR13	AREA TO SAVE R13
(C0)	192	SIGNED	4	LWOR14	AREA TO SAVE R14
(C4)	196	SIGNED	4	LWOR15	AREA TO SAVE R15
(C8)	200	BITSTRING	24	LWOSAVE	SAVE AREA FOR CALLER'S REGS.
(E0)	224	SIGNED	4	LWOXWA	ADDRESS TO EXIT WORK AREA
(E4)	228	SIGNED	2	LWOXWAL	LENGTH OF EXIT WORK AREA
(E6)	230	BITSTRING	2		RESERVED FOR FUTURE USE
(E8)	232	BITSTRING	20	LWOEXPL	OUTPUT EXIT PARAMETER LIST
LOGICAL/PHYSICAL WRITER COMMUNICATION INFO @DY43589					
(FC)	252	BITSTRING	4	LWAOUTLN	POINTER TO OUTPUT LINE
(100)	256	BITSTRING	2	LWNOIGNR	NUMBER OF IGNORED RECORDS
(100)	256		0	LWILN	"*-LADS" LENGTH OF CONTROL BLOCK

Message Control Block (MMB)

Definition Macro: IPW$DMM

This block provides support for the macros IPW$WTO and IPW$WTR. A routine issuing one of these macros will invoke message services. A message to be printed on SYSLOG will be passed to the MMB by means of the message request word in the TCB. The MMB also contains the channel program (CCB and CCW) to execute the I/O to the console. If a reply is necessary the channel program in the MMB will execute the necessary I/O. The message service will move the reply to an area addressed by the reply request word in the TCB for the task using the routine that issued the IPW$WTR macro. (See also TCMW and TCAW fields in the TCB.)

How to Locate: Refer to Figure 151 on page 730 in Chapter 6, “Diagnostic Aids.”

The format of this block as it is printed in a dump is as follows.

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
				MESSAGE CONTROL BLOCK (MMB)	
(0)	0	CHARACTER	16	MMSD	SECTION DESCRIPTOR
(10)	16	DBL WORD	8		ALLIGNMENT
(10)	16	BITSTRING	9	MMWW	WORK AREA
(19)	25	BITSTRING	3		UNUSED
(1C)	28	SIGNED	4	MMLK	LOCK WORD
(20)	32	CHARACTER	48	MMSV (0)	REGISTER SAVE AREA
(20)	32	SIGNED	4	MMRE	SAVED REGISTER 14
(24)	36	SIGNED	4	MMRF	SAVED REGISTER 15
(28)	40	SIGNED	4	MMR0	SAVED REGISTER 0
(2C)	44	SIGNED	4	MMR1	SAVED REGISTER 1
(30)	48	SIGNED	4	MMR2	SAVED REGISTER 2
(34)	52	SIGNED	4	MMR3	SAVED REGISTER 3
(38)	56	SIGNED	4	MMR4	SAVED REGISTER 4
(3C)	60	SIGNED	4	MMR5	SAVED REGISTER 5
(40)	64	SIGNED	4	MMR6	SAVED REGISTER 6
(44)	68	SIGNED	4	MMR7	SAVED REGISTER 7
(48)	72	SIGNED	4	MMR8	SAVED REGISTER 8
(4C)	76	SIGNED	4	MMR9	SAVED REGISTER 9
(50)	80	CHARACTER	16	MMCB (0)	COMMAND CONTROL BLOCK
(50)	80	SIGNED	2	MMCT	RESIDUAL COUNT
(52)	82	BITSTRING	2	MMCM	COMMUNICATION BYTES
(54)	84	BITSTRING	2	MMST	STATUS BYTES
(56)	86	BITSTRING	2	MMLU	LUB IDENTIFIER
(58)	88	BITSTRING	1	MMCA	FLAGS
(59)	89	ADDRESS	3		CHANNEL PROGRAM ADDRESS
(5C)	92	BITSTRING	4		DOS/VSE INTERNAL USE
(60)	96	CHARACTER	16	MMCH (0)	CHANNEL PROGRAM
(60)	96	CHARACTER	8	MMWT (0)	WRITE CCW
(60)	96	BITSTRING	1		WRITE COMMAND CODE
(61)	97	ADDRESS	3		DATA AREA ADDRESS
(64)	100	BITSTRING	2		FLAGS
(66)	102	ADDRESS	2		COUNT
(68)	104	CHARACTER	8	MMRD (0)	READ CCW
(68)	104	BITSTRING	1		READ COMMAND CODE
(69)	105	ADDRESS	3		DATA AREA ADDRESS
(6C)	108	BITSTRING	2		FLAGS

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(6E)	110	ADDRESS	2		COUNT
(70)	112	ADDRESS	2	MMMAL	MESSAGE LENGTH 1 FOR WTO
(72)	114	CHAR-ACTER	256	MMMA	MESSAGE OUTPUT AREA 1
(172)	370	CHAR-ACTER	72	MMMI	REPLY INPUT AREA
(1BA)	442	CHAR-ACTER	12		??? RESERVED
(1C8)	456	SIGNED	4	MMMSRET	RET'N REG. FOR IPW$$MS
(1CC)	460	ADDRESS	2	MMMBL	MESSAGE LENGTH 2 FOR WTO
(1CE)	462	CHAR-ACTER	70	MMMB	MESSAGE OUTPUT AREA 2
(214)	532	BITSTRING	1	MMTQE (0)	TIMER ELEMENT
(22C)	556	SIGNED	4	MMSRE	SAVEAREA FOR MMRE
(230)	560	SIGNED	4	MMSRF	SAVEAREA FOR MMRF
(234)	564	SIGNED	4	MMMSV (14)	FUNCTION SAVEAREA
		EXPRESSION		MMLN	"*-MMSD" LENGTH DESCRIPTOR

Master External Device Control Block

Definition Macro: IPW$DED

This control block is built at VSE/POWER initialization time and is used as anchor point for the external device control block (EDCB) chain. The control block is pointed to by field 'CAEDCB'. The format is as follows:

Bytes	Label	
Hex.	of Field	Description/Function
00	MEDCBDS	Start of DSECT
00-0F	MEDCBHD	Storage descriptor
10-13	MEDCBFEL	Address of first EDCB in chain
14-1B		Reserved for future use
1C-1F	MEDCBLW	Lockword

Module Control Block (MCB)

Definition Macro: IPW$DMC

Each module (an extent, always 1 for queue file and at least 1 for the data file) requires an MCB. The format and type of information contained in any MCB is identical.

The format of a module control block as it is printed in a dump is as follows.

Module Control Block for CKD Devices

Bytes	Label	
Hex.	of Field	Description/Function
00-0F	MCSD[1]	Storage descriptor MCB
10-17	MCSA	Module seek address (MBBCCHHR) (See Notes[2].
18	MCDT	Device type of file 0 = CKD device
19		Reserved for future use
1A-1B	MCQBM	Number of queue record blocks used for the Master Record
1C-1F	MCLK	Lockword
	MCCB	Command control block
20-21	MCCT	Residual count
22-23	MCCM	Communication bytes
24-25	MCST	Device status
26-27	MCLU	EXCP real plus LUB index (logical unit)
28-2B	MCCA	CCW address
2C-2F		CCW address in CSW
	MCXT	Extent information
30-33	MCLO	Low limit (CCHH)
34-37	MCHI	High limit (CCHH)
38-3B	MCFDB	Relative number of first DBLK in extent
3C-3F	MCLDB	Relative number of last DBLK in extent
40	MCSE	Sector value
41-43		Reserved for future use
44-47	MCSX	Sector table address
48-49	MC#R	Number of records per track
4A-4B	MC#T	Number of tracks per cylinder
4C-4F		Reserved for future use
	MCCH	Channel program
50-57	MCSK	Seek CCW
58-5F	MCSS	Set sector or TIC CCW
60-67	MCSH	Search CCW
68-6B	MCTI	TIC CCW
6C-6F	MCTV	Virtual address of buffer
70-77	MCRW	Read or write CCW
78-7B	MC$T	Owner of I/O request
7C-7F	MC$1	Save area for register 1 of current request
80-83	MCPNO	Queue record block number of previous I/O request
84-87	MCBF	Saved register 15 for disk service

Bytes	Label	
Hex.	of Field	Description/Function
88-8B	MCDA	Virtual address of I/O area
8C-8F	MCVI	Virtual address of IDAL list
90-9F	MCSV	Temporary save area

- The following area is used by the queue file server and the VIO move subroutine.

A0-AB	MCIOW	I/O request word
AC-B7	MCPRM	VIO move parameter list
AC-AF	MCPRB	Relative byte address in VIO
B0-B3	MCPVA	Address of storage area
B4-B5	MCPLL	Length of move operation
B6	MCPOP	Flag byte
	MCPIN	X'80' - move into VIO space
	MCPOU	X'40' - move out from VIO space
B7		Reserved for future use

Module Control Block for FBA Devices

00-0F	MFSD[1]	Storage descriptor MCB
10-17	MFLW	Locate word
18	MFDT	Device type of file F = FBA device
19		Reserved for future use
1A-1B	MFQBM	Number of queue record blocks used for the Master Record
1C-1F	MFLK	Lockword
20-2F	MFCB	Command control block
20-21	MFCT	Residual count
22-23	MFCM	Communication bytes
24-26	MFST	Device status
26-27	MFLU	EXCP real plus LUB index (logical unit)
28-2B	MFCA	CCW address
2C-2F		CCW address in CSW
	MFXT	Extent information
30-33	MFLO	Low limit (starting physical block number of extent)
34-37	MFHI	High limit (ending physical block number of extent)
38-3B	MFFDB	Relative number of first DBLK in extent
3C-3F	MFLDB	Relative number of last DBLK in extent
40-43	MFSI	Default block size for FBA devices
44-45	MFUT	Unit of transfer
46-47		Reserved for future use
	MFED	Extent description block

Bytes	Label	
Hex.	of Field	Description/Function
48	MFMB	Mask byte
		C0 = Permit all write commands
		04 = Permit all diagnostic commands
		40 = Inhibit all write commands
		44 = Inhibit all write commands and permit all diagnostic commands
49-4B		Reserved, must be zero
4C-4F	MFBB	Physical address of first block of extent
50-53	MFFB	Relative displacement of first block of extent
54-57	MFLB	Relative displacement of last block of extent
	MFCH	Channel program
58-5F	MFDF	Define extent CCW
60-67	MFLC	Locate CCW
68-6B	MFTI	TIC CCW
6C-6F	MFTV	Virtual address of buffer
70-77	MFRW	Read or write CCW
78-7B	MF$T	Owner of this I/O request
7C-7F	MF$1	Saved register 1 of current request
80-83	MFPNO	Queue record block number of previous I/O request
84-87	MFBF	Saved register 15 for disk service
88-8B	MFDA	Virtual address of data I/O area
8C-8F	MFVI	Virtual address of IDAL list
90-9F	MFSV	Temporary save area

Notes:

1. The labels in this control block vary according to the generated DSECT or declaration. The first characters are Q1 for the queue file MCB, D2 for the DFILE2 MCB, and MC for all other MCBs.
2. Seek and search address required by the channel program. Whenever an input or output operation is to be performed, this field is updated by the seek address evaluated from the relative data block or queue record number in the I/O request word.

How to Locate: Refer to Figure 152 on page 731 in Chapter 6, “Diagnostic Aids.”

Network Composer Work Area

Definition Macro: IPW$DWC

This work area is used by the composer to build records which it has received from the transmitter, into a transmission block.

Bytes	Label	
Hex.	of Field	Description/Function
000-07F		PLS dynamic data area
080-083	NCWAHDR	Work area header
084-087	NCWTCB	Points to task control block
088-095	NCWAPUT	Parameter area for PUT macro
088	NCWACOCO	TCB command code for current REC
089-08B	NCWARA	Address of record
08C	NCWAGP	Copied TCB gen. purpose byte
08D	NCWAG2	Copied TCB gen. purpose byte 2
	NCWAGJHR	X'80' - Job header record
	NCWAGJTR	X'40' - Job trailer record
	NCWAGDHR	X'20' - Dataset header record
	NCWAGPRI	X'10' - CPDS record(page record)
	NCWAGASA	X'02' - Record contains ASA control character
08E-08F	NCWARL	Length of record
090-091	NCWARLM	Maximum record length
092	NCWART	Record type passed over
093	NCWAMLI	MLI request
094	NCWRCB	RCB of task
095	NCWAIND	Various indications
	NCWAFXD	X'80' - Fixed format record indication
	NCWANOCC	X'40' - Record without carriage control
096	NCWFLAGS	Composer flag bits
	NCWLSEG	X'80' - Last segment indicator
	NCWSYSIN	X'40' - Input record indicator
	NCWSYSOU	X'20' - Output record indicator
	NCWNOCMP	X'10' - Do not compress indicator
	NCWASPR	- Spanned record indicators (indicates first/middle/last segment or UNSP)
	NCWFDSG	X'02' - First data segment indicator
	NCWEXP	X'01' - Blank expansion indicator
097		Reserved
098-09B	NCWWPTR	Points to first free byte in segment area work field
09C-09F	NCWNCROF	Offset pointer into NCR
0A0-0A3	NCWNCREA	Points to last byte of NCR
0A4-0A7	NCWSGPTR	Points to segment area NCWSGAR (initialized by transmitter)
0A8-0AB	NCWSGC	Segment counter
0AC-0AD	NCWSGLEN	Length of segment area (initialized by transmitter)
0AE	NCWRLTB	Length of blank string to be added to current record
0AF		Reserved

Bytes	Label	
Hex.	of Field	Description/Function
0B0-0B1	NCW#BLNK	No. of blanks to be expanded
0B2-0B3	NCWTBLEN	Length of NJE transmission block
0B4-0B5	NCWRED	Reduction variable for XMIT block size
0B6-0B7	NCWRCL	Length of preprec input record
0B8-0BB	NCWRCA	Addr. of preprec input record
0BC-0BF	NCWINA	Addr. of DATBUF input record
0C0-0C1	NCWINL	Length of DATBUF input record
0C2-0C3		Reserved
0C4-0C7	NCWBUFA	Address of output buffer
0C8-0CB	NCWBUFOF	Offset in output buffer
0CC-0CD	NCWBUFRL	Residual output buffer length
0CE-0CF	NCWOUTRL	Length of complete record to be put into output buffer (incl. RCB, SRCB, etc., EOB-RCB)
0D0-0D2	NCWGETBF	Operand field of GETBUF call
0D3		Reserved
0D4-0D7	NCWSRCBP	Pointer to actual SRCB field
0D8-0DB	NCWKOPTR	Points to compression output area (initialized by transmitter)
0DC-0DD	NCWKOL	Length of compression output area (initialized by transmitter)
0DE	NCWSTAT	Saves compression error status
0DF		Reserved
0E0-0E3	NCWHDR	Data record header
0E0-0E1	NCWHDR1	If first segment: total length of spanned record else: length of segment
0E0	NCWHDR11	If unspanned record: length of record
0E1	NCWHDR12	If unspanned record: command code
0E2	NCWHDR2	First segment: Length of segment
0E3	NCWHDR3	First segment: command code
0E4	NCWHDRL	Length of record header
0E5		Reserved
0E6-0E7	NCWRESL	Part of DATREC not yet processed
0E8-11F	NCWFSVE	Composer save area
120-143	NCDKABLK	Storage for compression work area

Network Compression Work Area

Definition Macro: IPW$DKA

This work area is used by the compression routine when it is compressing records before building them into a transmission buffer. It is also used by the decompression routine when it decompresses buffers which it has received from the PNET driver.

Offset Hex	Type	Len	Name (Dim)	Description
(0)	ADDRESS	4	DKAINFA	ADDR. OF INPUT FIELD (STRING)
(4)	ADDRESS	4	DKAINFEA	INPUT FIELD END ADDRESS + 1
(8)	ADDRESS	4	DKAOUFA	ADDRESS OF OUTPUT FIELD
(C)	SIGNED	2	DKAOUFLN	LENGTH OF OUTPUT FIELD
(E)	SIGNED	2	DKAOULN	OUTPUT STRING LENGTH
(10)	ADDRESS	4	DKAOUFEA	ADDR. LAST BYTE OF OUTPUT FIELD + 1
(14)	BITSTRING	1	DKAREQ	REQUEST BYTE:
	....		DKABCR	"X'00'" - COMPRESSION, BSC MODE
	.... .1..		DKABDCR	"X'04'" - DECOMPRESSION, BSC MODE
	...1		DKASCR	"X'10'" - COMPRESSION, SNA MODE
	...1 .1..		DKASDCR	"X'14'" - DECOMPRESSION INCLUDING DECOMPACTION, SNA MODE
	...1 1...		DKASNEEK	"X'18'" - DECOMPRESSION, SNA MODE(SNEEK-A-PEEK)
(15)	BITSTRING	1	DKASTAT	OUTPUT STATUS BYTE:
	....		DKASTAT0	"X'00'" - NO ERROR OCCURRED
	1		DKAERR01	"X'01'" - OUTPUT STRING EXCEEDS OUTPUT AREA
	.... ..1.		DKAERR02	"X'02'" - SCB ERROR. LENGTH EXCEEDS INPUT AREA.
	11		DKAERR03	"X'03'" - INVALID REQUEST CODE
	.... .1..		DKAERR04	"X'04'" - LENGTH OF INPUT STRING <= 0.
	.... .1.1		DKAERR05	"X'05'" - LENGTH OF OUTPUT FIELD <= 0.
	.... .11.		DKAERR06	"X'06'" - INVALID SCB ENCOUNTERED
	.... .111		DKAERR07	"X'07'" - ERROR, SCB COUNT = 0
	.... 1..1		DKAERR09	"X'09'" - OUTPUT LENGTH EXCEEDS OUTPUT BNDY
	.... 1.1.		DKAERR0A	"X'0A'" - SNA DECOMP.: END OF INPUT OCCURRED, BEFORE REQUESTED OUTPUT LNGTH REACHED
	.... 1.11		DKAERR0B	"X'0B'" - NO COMPACTION TABLE AVAIL.
(16)	BITSTRING	1	DKAFLAG	FLAG BYTE
	1...		DKAHALF	"X'80'" HALF BYTE DECOMPACTED
(17)	CHARACTER	1		RESERVED
(18)	ADDRESS	4	DKANSCB	POINT TO START SCB (SNA ONLY)
(1C)	ADDRESS	4	DKAINPOS	POINTS TO FIRST BYTE, NOT YET PROCESSED, IN INPUT STRING
(20)	SIGNED	2	DKAINFLN	LENGTH OF INPUT FIELD (STRING)
(22)	SIGNED	2	DKAREQL	SNA OUTPUT BUFFER LENGTH (BYTES)
(24)	ADDRESS	4	DKACMPT	ADDR. OF COMPACTION TABLE BLOCK (SNA ONLY)

Network Definition Table (NDT)

Definition Macro: PNODE DSECT=YES

The Network Definition Table (NDT) is used to define the network environment to PNET. It is generated from user definitions made with the PNODE macro. There must be one entry in the table for every node which must be known by the node on which this table will be used. The table is loaded into the PNET environment by use of the PLOAD PNET= command, or is automatically done during initialization, if PNET= was specified in the POWER generation.

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
NETWORK DEFINITION TABLE: HEADER					
(0)	0	STRUCTURE	0	NTHDS	, HEADER DEFINITION DSECT
(0)	0	CHARACTER	16	NTHSD	STORAGE DESCRIPTOR
(10)	16	ADDRESS	4	NTHFE	ADDRESS OF FIRST ENTRY IN TABLE
(14)	20	SIGNED	2	NTHNE	NUMBER OF ENTRIES IN TABLE
(16)	22	SIGNED	2	NTHMX	MAX. # TCP/IP SOCKET CALLS
(18)	24	ADDRESS	4	NTHOE	ADDRESS OF OWN ENTRY IN TABLE
(1C)	28	CHARACTER	4	NTHVM	NETWORK DEF. TABLE VERS. ID
(20)	32	BITSTRING	2		RESERVED FOR FUTURE USE
(22)	34	CHARACTER	44		COPYRIGHT INFORMATION
(4E)	78	BITSTRING	50		RESERVED FOR FUTURE USE
		1...		NTHLN	"*-NTHDS" LENGTH OF HEADER

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
NETWORK DEFINITION TABLE: ENTRIES					
(0)	0	STRUCTURE	0	NDTDS	, NETWORK DEFINITION TABLE DSECT
(0)	0	CHARACTER	8	NDTNM	NODE NAME
(8)	8	CHARACTER	8	NDTPW	PASSWORD
(10)	16	BITSTRING	1	NDTAF	AUTHORITY FLAGS
		1111		NDTAS	"X'F0'" .. NODE IS 'SYSTEM' AUTHORIZED
		.1.1		NDTAN	"X'50'" .. NODE IS 'NETWORK' AUTHORIZED
		..1.		NDTNU	"X'20'" .. NOT USED
		...1		NDTAJ	"X'10'" .. NODE IS 'JOB' AUTHORIZED
		....		NDTAE	"X'00'" .. NODE IS NOT AUTHORIZED AT ALL
(11)	17	BITSTRING	2		RESERVED FOR FUTURE USE
(13)	19	BITSTRING	1	NDTF1	FLAG BYTE 1
		1...		NDTLI	"X'80'" .. ADJACENT BSC/CTC NODE
		.1..		NDTVA	"X'40'" .. SNA TYPE NODE MAY BE SET FOR OWN NODE!
		..1.		NDTF1IA	"X'20'" .. INVALID SNA APPLID
		...1		NDTPA	"X'10'" .. TCP TYPE NODE
		.... 1...		NDTF1IP	"X'08'" .. INVALID TCP ADDRESS
		.... .1..		NDTSA	"X'04'" .. SSL TYPE NODE
(14)	20	CHARACTER	8	NDTPR	PRIME ROUTE FOR BSC/CTC, OR APPLICATION ID FOR SNA, OR '*ATP ', IF TCP IPHOSTAD '*ATP ', IF SSL ISHOSTAD '*NTP ', IF TCP IPHOSTNM '*NTP ', IF SSL ISHOSTNM
(1C)	28	CHARACTER	8	NDTSR	ALTERNATE ROUTE
(24)	36	ADDRESS	2	NDTBS	PNET BUFFERSIZE
(26)	38	ADDRESS	1	NDTNI	MAX. NO OF INPUT BUFFERS

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(27)	39	ADDRESS	1	NDTNO	MAX. NO OUTPUT BUFFERS
(28)	40	ADDRESS	2	NDTPT	TCP/IP PORT NUMBER FOR ... 'CONNECT', IF REMOTE NODE OR 'LISTEN', IF OWN NODE
(2A)	42	ADDRESS	2	NDTSPT	TCP/IP SSL PORT NUMBER FOR.. 'CONNECT', IF REMOTE NODE OR 'LISTEN', IF OWN NODE
(2C)	44	ADDRESS	4	NDTHML	TCP, LENGTH OF NDTHM VALUE
(30)	48	SIGNED	4	NDTIPAD (0)	BINARY IP ADDRESS
(30)	48	ADDRESS	1	NDTIPA1	BINARY IPADDR BYTE 1| FOR...
(31)	49	ADDRESS	1	NDTIPA2	BINARY IPADDR BYTE 2| DOTTED
(32)	50	ADDRESS	1	NDTIPA3	BINARY IPADDR BYTE 3| DECIM.
(33)	51	ADDRESS	1	NDTIPA4	BINARY IPADDR BYTE 4| VALUE
(34)	52	CHARACTER	255	NDTHM	TCP/IP HOST NAME ... FOR OWN NODE 'UNUSED', FOR REMOTE NODE ... DOTTED DECIMAL (NDTPR=*ATP) SYMBOLIC NAME, (NDTPR=*NTP)
(133)	307	BITSTRING	1	NDTF2	FLAG BYTE 2
		1...		NDT2CRL	"X'80'" .. ENCRYPT = LOW
		.1..		NDT2CRH	"X'40'" .. ENCRYPT = HIGH
(134)	308	SIGNED	4	NDTMSGTM	TIME STAMP FOR MSG 1RC6
(138)	312	SIGNED	4	NDTMSGDI	DOM-ID FOR A-TYPE MSG
(13C)	316	SIGNED	4		RESERVED FOR FUTURE USE
(140)	320	CHARACTER	8	NDTSSLT	SSL TYPE OF SECURITY PROTOC.
(148)	328	CHARACTER	16	NDTKEYR	LIB.SUBLIB SSL KEY DATABASE
(158)	344	CHARACTER	8	NDTKEYM	KEY MEMBER IN KEY DATABASE
(160)	352	SIGNED	4	(7)	RESERVED FOR FUTURE USE
(160)	352		0	NDTLN	"*-NDTDS" LENGTH OF ENTRY

Network Data Set Header Record (DSHR)

Definition Macro: IPW$DNR DHR=YES

The data set header record is a control record which is normally only present on output data sets (list or punch). It contains information relevant to the output, e.g. forms number, output class. A short form of the record, the record characteristics change section, may also be present on input jobs, if the record length of the data set is not 80 bytes.

The layout of the first four bytes of every header record is identical. Bytes 0 and 1 are the length of the entire block. Individual records must not be greater than 256 bytes long, so if the total record is longer it must be segmented. Byte 2 is a flag byte that is zero. Byte 3 is the transmission sequence indicator and is used to indicate that a header has been segmented. The high order bit (X'1... ....') indicates that there are more parts to come, and the other bits are a sequence counter of the blocks for this record. For example, if the header had to be split into three parts then the sequence indicators in the three blocks would be as follows:- X'80', X'81', and X'02'.

The layout of the first four bytes of all sections is always identical. The section flags are to be found in the description of the job header record (refer to “Network Job Header Record (JHR)” on page 559).

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
Network Data Set Header Record (DSHR)					
(0)	0	STRUCTURE		NDHDSECT	NETWORK DATA SET HEADER RECORD
BLOCK CONTROL INFORMATION					
(0)	0	ADDRESS	2	NDHLEN	LENGTH OF ENTIRE BLOCK
(2)	2	BITSTRING	1	NDHFLAGS	FLAGS
(3)	3	BITSTRING		NDHSEQ	TRANSMISSION SEQUENCE INDICATOR
		.... .1..		NDHLBCI	"*-NDHDSECT" LENGTH OF BLOCK CONTROL INFORMAT
GENERAL SECTION					
(4)	4	SIGNED	4	NDHG (0)	START OF GENERAL SECTION
(4)	4	ADDRESS	2	NDHGLEN	LENGTH OF GENERAL SECTION
(6)	6	BITSTRING	2	NDHGFLGS (0)	SECTION TYPE FLAGS
(6)	6	ADDRESS	1	NDHGTYPE	ID FOR GENERAL SECTION
(7)	7	ADDRESS	1	NDHGMOD	MODIFIER
		....		NDHG$MOD	"B'00000000'" .. VALUE OF MODIFIER
(8)	8	CHARACTER	8	NDHGNODE	DESTINATION NODE NAME
(10)	16	CHARACTER	8	NDHGRMT	DESTINATION REMOTE NAME
(18)	24	CHARACTER	8	NDHGPROC	PROC INVOCATION NAME
(20)	32	CHARACTER	8	NDHGSTEP	STEP NAME
(28)	40	CHARACTER	8	NDHGDD	DD NAME
(30)	48	SIGNED	2	NDHGDSNO	DATA SET NUMBER
(32)	50	ADDRESS	1		RESERVED
(33)	51	CHARACTER	1	NDHGCLAS	OUTPUT CLASS
(34)	52	SIGNED	4	NDHGNREC	RECORD COUNT
(38)	56	BITSTRING	1	NDHGFLG1	FLAGS
		1...		NDHGF1SP	"B'10000000'" .. SPIN DATA SET (SEGMENTED)
		.1..		NDHGF1HD	"B'01000000'" .. HOLD DATA SET AT DESTINATION
		..1.		NDHGF1LG	"B'00100000'" .. JOB LOG INDICATOR
		...1		NDHGF1OV	"B'00010000'" .. PAGE OVERFLOW INDICATOR

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
		.... 1...		NDHGF1IN	"B'00001000'" .. PUNCH INTERPRET INDICATOR
(39)	57	BITSTRING	1	NDHGRCFM	RECFM
		11..		NDHGRCUN	"B'11000000'" .. UNDEFINED FORMAT
		1...		NDHGRCFF	"B'10000000'" .. FIXED FORMAT
		.1..		NDHGRCVF	"B'01000000'" .. VARIABLE FORMAT
		.... .1..		NDHGRCAS	"B'00000100'" .. ASA CONTROL CHARACTERS
		.... ..1.		NDHGRCMC	"B'00000010'" .. MACHINE CNTRL CHARACTER
(3A)	58	SIGNED	2	NDHGLREC	MAX LOGICAL RECORD LENGTH
(3C)	60	ADDRESS	1	NDHGDSCT	DATA SET COPY COUNT
(3D)	61	ADDRESS	1	NDHGFCBI	3211 FCB INDEX
(3E)	62	ADDRESS	1	NDHGLNCT	DEFAULT LINES PER PAGE
(3F)	63	BITSTRING	1		RESERVED
(40)	64	CHARACTER	8	NDHGFORM	FORMS ID
(48)	72	CHARACTER	8	NDHGFCB	FCB ID
(50)	80	CHARACTER	8	NDHGUCS	UCS ID
(58)	88	CHARACTER	8	NDHGXWTR	EXTERNAL WRITER ID
(60)	96	CHARACTER	8		RESERVED
(68)	104	BITSTRING	1	NDHGFLG2	SECOND FLAG BYTE
		1...		NDHGF2PR	"X'80'" .. DATASET IS TO BE PRINTED
		.1..		NDHGF2PU	"X'40'" .. DATASET IS TO BE PUNCHED
		..1.		NDHGF2HB	"X'20'" .. HOLD DATASET BEFORE
		...1		NDHGF2HA	"X'10'" .. HOLD DATASET AFTER
(69)	105	BITSTRING	1	NDHGUCSO	UCS OPTION BYTE
		1...		NDHGUCSD	"X'80'" .. UCS DATA CHECK OPTION
		.1..		NDHGUCSF	"X'40'" .. UCS FOLDING REQUESTED OPTION
(6A)	106	BITSTRING	2		RESERVED
(6C)	108	CHARACTER	8	NDHGPMDE	PROCESS MODE
(74)	116	SIGNED	4	NDHGEND (0)	END OF GENERAL SECTION
		.111		NDHGLLEN	"*-NDHG" LENGTH OF GENERAL SECTION

VSE/POWER SUBSYSTEM SECTION (LONG FORM)

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(74)	116	SIGNED	4	NDHP (0)	START OF VSE/POWER SECTION
(74)	116	ADDRESS	2	NDHPLEN	LENGTH OF VSE/POWER SECTION
(76)	118	BITSTRING	2	NDHPFLGS (0)	SECTION TYPE FLAGS
(76)	118	ADDRESS	1	NDHPTYPE	ID FOR VSE/POWER SECTION
(77)	119	ADDRESS	1	NDHPMOD	MODIFIER
		....		NDHP$MOD	"B'00000000'" .. VALUE OF MODIFIER
		1		NDHP$MD1	"B'00000001'" .. VALUE OF MODIFIER SHORT FORM
(78)	120	BITSTRING	1	NDHPFLG1	FLAGS
		1...		NDHPF1AC	"X'80'" .. CREATED BY CMD 'J PUN'
(79)	121	BITSTRING	1	NDHPIDEV	DOS/VSE DEVICE TYPE
(7A)	122	CHARACTER	1	NDHPPRIO	OUTPUT PRIORITY
(7B)	123	CHARACTER	1	NDHPDISP	OUTPUT DISPOSITION
(7C)	124	CHARACTER	16	NDHPUSER	USER INFORMATION
(8C)	140	BITSTRING	1	NDHPJBSF	JOB SUFFIX
		1...		NDHPJBLA	"X'80'" .. LAST SEGMENT INDICATOR (NOTE: BITS 1 - 7 ARE THE JOB SUFFIX NUMBER(1 - 127) IF ANY)
(8D)	141	CHARACTER	1	NDHPSYID	SYSTEM ID
(8E)	142	ADDRESS	1	NDHPNSEP	NUMBER OF SEPARATOR PAGES
(8F)	143	BITSTRING	1	NDHPOPTN	GENERAL OPTION BYTE 1 (NOTE - BITS ARE DEFINED IN DMB AND IN QUEUE RECORD)
		..1.		NDHPCSUP	"X'20'" .. NO SEP PAGES BTWN COPIES

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
		...1		NDHPOPHP	"X'10'" .. HOLD IF PRT/PUN FAILS
(90)	144	CHAR-ACTER	2	NDHPPART	DOS/VSE PARTITION ID
(92)	146	ADDRESS	2		RESERVED
(94)	148	BITSTRING	1	NDHPRCFM	SPECIAL RECORD FORMAT
		1...		NDHPRCSC	"X'80'" .. SCS PRINT FORMAT
		.1..		NDHPRCBM	"X'40'" .. BMS MAPPING FORMAT
		..1.		NDHPRC32	"X'20'" .. 3270 RECORD FORMAT
		...1		NDHPRCAP	"X'10'" .. APA DATA FORMAT (CPDS)
		.... 1...		NDHPRCES	"X'08'" .. ESCAPE MODE FORMAT
		.... .1..		NDHPRCAS	"X'04'" .. ASA CARRIAGE CONTROL
		.... ..1.		NDHPRCMC	"X'02'" .. MACHINE CARRIAGE CNTRL.
(95)	149	CHAR-ACTER	1		UNUSED
(96)	150	ADDRESS	2	NDHPJNUM	JOB NO OF OUTPUT SEGMENT
(98)	152	CHAR-ACTER	4	NDHPCOMP	COMPACTION TABLE NAME
(9C)	156	CHAR-ACTER	8	NDHPPASS	PASSWORD
(A4)	164	CHAR-ACTER	68	NDHPSETP	SETPRT PARAMETER LIST
(E8)	232	SIGNED	8	NDHPSTRT	START TIME/DATE FOR JTR
(F0)	240	SIGNED	4	NDHPEND (0)	END OF VSE/POWER SECTION
		.111 11..		NDHPLLEN	"*-NDHP" LENGTH OF VSE/POWER SECTION

VSE/POWER SUBSYSTEM SECTION (SHORT FORM)

This form is used only on input, whenever data is read from a IBM 3540 diskette device.

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(79)	121	BITSTRING	1		UNUSED
(7A)	122	BITSTRING	2	NDHPCUU	3540 CUU
		.... 1...		NDHPCLEN	"*-NDHP" LENGTH OF SHORT VSE/POWER SECTION

3800 PRINTER CHARACTERISTICS GENERAL SECTION (OPTIONAL)

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(F0)	240	SIGNED	4	NDHA (0)	START OF 3800 CHAR SECTION
(F0)	240	ADDRESS	2	NDHALEN	LENGTH OF 3800 CHAR SECTION
(F2)	242	BITSTRING	2	NDHAFLGS (0)	FLAGS AND MODIFIER
(F2)	242	ADDRESS	1	NDHATYPE	ID FOR GENERAL SECTION
(F3)	243	ADDRESS	1	NDHAMOD	MODIFIER
		1...		NDHA$MOD	"B'10000000'" .. VALUE OF MODIFIER (3800 CHAR)
(F4)	244	BITSTRING	1	NDHAFLG1	FLAGS
		1...		NDHAF1J	"B'10000000'" .. 'OPTCD=J' SPECIFIED
		.1..		NDHAF1BR	"B'01000000'" .. 'BURST=YES' SPECIFIED
		..1.		NDHAF1BN	"B'00100000'" .. 'BURST=NO' SPECIFIED
(F5)	245	ADDRESS	1	NDHAFLCT	FLASH COUNT
(F6)	246	BITSTRING	1	NDHATREF	TABLE REFERENCE CHARACTER
(F7)	247	BITSTRING	1		RESERVED
(F8)	248	CHAR-ACTER	8	NDHATAB1	TRANSLATE TABLE 1
(100)	256	CHAR-ACTER	8	NDHATAB2	TRANSLATE TABLE 2
(108)	264	CHAR-ACTER	8	NDHATAB3	TRANSLATE TABLE 3
(110)	272	CHAR-ACTER	8	NDHATAB4	TRANSLATE TABLE 4
(118)	280	CHAR-ACTER	8	NDHAFLSH	FLASH CARTRIDGE ID
(120)	288	CHAR-ACTER	8	NDHAMODF	COPY MODIFICATION ID
(128)	296	BITSTRING	8	NDHACPYG	COPY GROUPS
(130)	304	SIGNED	4	NDHAEND (0)	END OF 3800 CHAR SECTION
		.1..		NDHALLEN	"*-NDHA" LENGTH OF 3800 CHAR SECTION
(130)	304			NDHLLEN	"*-NDHDSECT" LENGTH OF ENTIRE BLOCK

RECORD CHARACTERISTICS CHANGE GENERAL SECTION

This section is used only within an input data stream when the record length is not 80 bytes.

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(130)	304	SIGNED	4	NDHC (0)	START OF CHAR CHANGE GENERAL SECTI
(130)	304	ADDRESS	2	NDHCLEN	LENGTH OF CHAR CHANGE GEN SECT
(132)	306	BITSTRING	2	NDHCFLGS (0)	SECTION TYPE FLAGS
(132)	306	ADDRESS	1	NDHCTYPE	ID FOR GENERAL SECTION
(133)	307	ADDRESS	1	NDHCMOD	MODIFIER
		.1..		NDHC$MOD	"B'01000000'" .. VALUE OF MODIFIER (CHAR CHANG
(134)	308	BITSTRING	1	NDHCFLG1	FLAGS
(135)	309	BITSTRING	1	NDHCRCFM	RECFM
(136)	310	ADDRESS	2	NDHCLREC	MAXIMUM LRECL
(138)	312	SIGNED	4	NDHCEND (0)	END OF CHAR CHANGE GENERAL SECT
		.... 1...		NDHCLLEN	"*-NDHC" LENGTH OF CHAR CHANGE GEN SECT

- VSE/POWER Output Processing Section

This section is built whenever Output Parameters must be processed by VSE/POWER.

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
Output Processing Section					
(0)	0	STRUCTURE		NDHS	, START OUTPUT PROC SECTION
(0)	0	ADDRESS	2	NDHSLEN	LENGTH OF SECTION
(2)	2	BITSTRING	2	NDHSFLGS (0)	FLAGS AND MODIFIER
(2)	2	ADDRESS	1	NDHSTYPE	- ID FOR GENERAL SECTION
(3)	3	ADDRESS	1	NDHSMOD	- MODIFIER
		....		NDHS$OUT	"B'00000000'" .. MODIFIER (OUTPUT)
(4)	4	ADDRESS	2	NDHSFLEN	SUBSECTION FIXED LENGTH
(6)	6	BITSTRING	1	NDHSFLG1	DATA STREAM FLAGS
		1...		NDHSCPDS	"B'10000000'" ..DATASET HAS CPDS RECORDS
(7)	7	BITSTRING	1		RESERVED
(8)	8	BITSTRING	8	NDHSJDVT	JDVT NAME
(10)	16	BITSTRING	4	NDHSNSTR	PAGE DATA SET PAGE COUNT
(14)	20	CHARACTER	8	NDHSGPID	OUTPUT GROUP NAME
		...1 11..		NDHSLEN2	"*-NDHS" LENGTH OF FIXED PART
(1C)	28	SIGNED	2	NDHSOPTB (0)	START OF OPTB DATA
(1C)	28	CHARACTER	4	NDHSPRID	PREFIX IDENTIFIER
(20)	32	ADDRESS	1	NDHSVERS	PREFIX VERSION LEVEL
(21)	33	ADDRESS	1	NDHSPLEN	LENGTH OF PREFIX
(22)	34	ADDRESS	2	NDHSDLEN	DATA LENGTH FOLLOWING PRFX
(24)	36	CHARACTER	8	NDHSVERB	'OUTPUT' CONSTANT
(2C)	44	CHARACTER	8	NDHSVRBL	
(34)	52	BITSTRING	1	NDHSFLG2	FLAG BYTE
		1...		NDHSCONT	"X'80'" .. OTHER OPTB STRUC EXISTS
(35)	53	ADDRESS	1	NDHSPARM	NO OF PARAMETERS PROCESSED
(36)	54	BITSTRING	2	NDHSRSV1	RESERVED
(38)	56	CHARACTER	1	NDHSTEXT (0)	START OF OPTB TEXT
		..11 1...		NDHSLEN1	"*-NDHS" LENGTH OUTPUT PROC SECTION

Network Job Header Record (JHR)

Definition Macro: IPW$DNR JHR=YES

The Job Header record (JHR) is a control record which, together with the job trailer record, is used as the 'bounds' for jobs or output which are to be transmitted via the network. It contains several different sections and only those relevant to VSE/POWER are described here.

The layout of the first four bytes of every header record is identical. Bytes 0 and 1 are the length of the entire block. Individual records must not be greater than 256 bytes long, so if the total record is longer it must be segmented. Byte 2 is a flag byte that is zero. Byte 3 is the transmission sequence indicator and is used to indicate that a header has been segmented. The high order bit (X'1... ....') indicates that there are more parts to come, and the other bits are a sequence counter of the blocks for this record. For example, if the header had to be split into three parts then the sequence indicators in the three blocks would be as follows:- X'80', X'81', and X'02'.

The layout of the first four bytes of all sections is always identical, so that it is easy to bypass sections for which there is no interest.

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
					NJE JOB HEADER RECORD
	BLOCK CONTROL INFORMATION				
(0)	0	ADDRESS	2	NJHLEN	LENGTH OF ENTIRE BLOCK
(2)	2	BITSTRING	1	NJHFLAGS	FLAGS
(3)	3	BITSTRING		NJHSEQ	TRANSMISSION SEQUENCE INDICATOR
		.... .1..		NJHLBCI	"*-NJHDSECT" LENGTH OF BLOCK CONTROL INFORMAT
	GENERAL SECTION				
(4)	4	SIGNED	4	NJHG (0)	START OF GENERAL SECTION
(4)	4	ADDRESS	2	NJHGLEN	LENGTH OF GENERAL SECTION
(6)	6	BITSTRING	2	NJHGFLGS (0)	SECTION TYPE FLAGS
(6)	6	ADDRESS	1	NJHGTYPE	ID FOR GENERAL SECTION
(7)	7	ADDRESS	1	NJHGMOD	MODIFIER
		....		NJHG$MOD	"B'00000000'" .. VALUE OF MODIFIER
(8)	8	ADDRESS	2	NJHGJID	JOB IDENTIFIER (NUMBER)
(A)	10	CHARACTER	1	NJHGJCLS	JOB CLASS
(B)	11	CHARACTER	1	NJHGMCLS	MESSAGE CLASS
(C)	12	BITSTRING	1	NJHGFLG1	FLAGS
		1...		NJHGF1PR	"B'10000000'" .. DO NOT RECOMPUTE PRIORITY
(D)	13	ADDRESS	1	NJHGPRIO	SELECTION PRIORITY
(E)	14	ADDRESS	1	NJHGORGQ	ORIGIN NODE SYSTEM QUALIFIER
(F)	15	ADDRESS	1	NJHGJCPY	JOB COPY COUNT
(10)	16	ADDRESS	1	NJHGLNCT	JOB LINE COUNT
(11)	17	BITSTRING	3		RESERVED
(14)	20	CHARACTER	8	NJHGACCT	NETWORKING ACCOUNT NUMBER
(1C)	28	CHARACTER	8	NJHGJNAM	JOB NAME
(24)	36	CHARACTER	8	NJHGUSID	USER ID (TSO, VM, ICCF) FOR NTFY
(2C)	44	CHARACTER	8	NJHGPASS	PASSWORD
(34)	52	SIGNED	8	NJHGNPAS	NEW PASSWORD
(3C)	60	SIGNED	8	NJHGETS	ENTRY TIME/DATE STAMP
(44)	68	CHARACTER	8	NJHGORGN	ORIGIN NODE NAME

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(4C)	76	CHARACTER	8	NJHGORGR	ORIGIN REMOTE NAME
(54)	84	CHARACTER	8	NJHGXEQN	EXECUTION NODE NAME
(5C)	92	CHARACTER	8	NJHGXEQU	EXECUTION USER ID(VM/370)
(64)	100	CHARACTER	8	NJHGPRTN	DEFAULT PRINT NODE NAME
(6C)	108	CHARACTER	8	NJHGPRTR	DEFAULT PRINT REMOTE NAME
(74)	116	CHARACTER	8	NJHGPUNN	DEFAULT PUNCH NODE NAME
(7C)	124	CHARACTER	8	NJHGPUNR	DEFAULT PUNCH REMOTE NAME
(84)	132	CHARACTER	8	NJHGFORM	JOB FORMS
(8C)	140	SIGNED	4	NJHGICRD	INPUT CARD COUNT
(90)	144	SIGNED	4	NJHGETIM	ESTIMATED EXECUTION TIME
(94)	148	SIGNED	4	NJHGELIN	ESTIMATED OUTPUT LINES
(98)	152	SIGNED	4	NJHGECRD	ESTIMATED OUTPUT CARDS
(9C)	156	CHARACTER	20	NJHGPRGN	PROGRAMMERS NAME
(B0)	176	CHARACTER	8	NJHGROOM	PROGRAMMERS ROOM NUMBER
(B8)	184	CHARACTER	8	NJHGDEPT	PROGRAMMERS DEPT NUMBER
(C0)	192	CHARACTER	8	NJHGBLDG	PROGRAMMERS BLDG NUMBER
(C8)	200	SIGNED	4	NJHGNREC	RECORD COUNT ON OUTPUT XMISSION
(CC)	204	SIGNED	4	NJHGJNO	JOB NUMBER IF > 64K
(D0)	208	CHARACTER	8	NJHGNTYN	NOTIFY NODE NAME
(D8)	216	SIGNED	4	NJHGEND (0)	END OF GENERAL SECTION
		11.1 .1..		NJHGLLEN	"*-NJHG" LENGTH OF GENERAL SECTION
	VSE/POWER SUBSYSTEM SECTION				
(D8)	216	SIGNED	4	NJHP (0)	START OF VSE/POWER SECTION
(D8)	216	ADDRESS	2	NJHPLEN	LENGTH OF VSE/POWER SECTION
(DA)	218	BITSTRING	2	NJHPFLGS (0)	SECTION TYPE FLAGS
(DA)	218	ADDRESS	1	NJHPTYPE	ID FOR VSE/POWER SECTION
(DB)	219	ADDRESS	1	NJHPMOD	MODIFIER
		....		NJHP$MOD	"B'00000000'" .. VALUE OF MODIFIER
(DC)	220	BITSTRING	1	NJHPFLG1	FLAGS
(DD)	221	CHARACTER	1	NJHPDISP	JOB DISPOSITION
(DE)	222	BITSTRING	1		RESERVED
(DF)	223	CHARACTER	1	NJHPSYID	TARGET SYSTEM IDENTIFIER
(E0)	224	CHARACTER	16	NJHPUSER	ORIGINATORS USER INFORMATION
(F0)	240	BITSTRING	2	NJHPDSKT	3540 SYSIN CUU
(F2)	242	BITSTRING	2		UNUSED
(F4)	244	BITSTRING	15	NJHPDD (0)	DUE DATE INFORMATION
(F4)	244	BITSTRING	1	NJHPDGP1	GENERAL PURPOSE BYTE1
		1...		NJHPDG1R	"X'80'" RERUN=YES SPECIFIED
					X'40' RESERVED FOR DUE INFO
					X'20' RESERVED FOR DUE INFO
					X'10' RESERVED FOR DUE INFO
		.... 1...		NJHPDG1T	"X'08'" DAILY SPECIFIED
		.... .1..		NJHPDG1W	"X'04'" WEEKDAYS SPECIFIED
		.... ..1.		NJHPDG1D	"X'02'" DAYS SPECIFIED
					X'01' RESERVED FOR DUE INFO
(F5)	245	BITSTRING	1	NJHPDGP2	GENERAL PURPOSE BYTE2
		1...		NJHPD280	"X'80'" RESERVED FOR DUE INFO
		.1..		NJHPD240	"X'40'" RESERVED FOR DUE INFO

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
		..1.		NJHPD220	"X'20'" RESERVED FOR DUE INFO
		...1		NJHPD210	"X'10'" RESERVED FOR DUE INFO
		.... 1...		NJHPD208	"X'08'" RESERVED FOR DUE INFO
		.... .1..		NJHPD204	"X'04'" RESERVED FOR DUE INFO
		.... ..1.		NJHPD202	"X'02'" RESERVED FOR DUE INFO
		1		NJHPDG2X	"X'01'" DUE DATE INFO EXISTS
(F6)	246	BITSTRING	6	NJHPDCY (0)	START OF CYCLING INFO
(F6)	246	BITSTRING	2	NJHPDMY	MONTHS WITHIN YEAR X'80' LEFT ALIGNED: 80=JAN X'40' 40=FEB, 20=MAR, ...
(F8)	248	BITSTRING	4	NJHPDDM	DAYS WITHIN MONTH BR;X'80' LEFT ALIGNED: 80=1ST BR;X'40' 40=2ND, 20=3RD, ...
(FC)	252	BITSTRING	6	NJHPDN (0)	NEXT DUE DATE/TIME IN PACKED DEC WITHOUT SIGN
(FC)	252	BITSTRING	4	NJHPDNDT (0)	NEXT DUE DATE (W/O TIME)
(FC)	252	BITSTRING	2	NJHPDNY	YEAR (1988-2087)
(FE)	254	BITSTRING	1	NJHPDNM	MONTH (1-12)
(FF)	255	BITSTRING	1	NJHPDND	DAY (1-31)
(100)	256	BITSTRING	2	NJHPDNT (0)	TIME
(100)	256	BITSTRING	1	NJHPDNTH	HOURS (0-23)
(101)	257	BITSTRING	1	NJHPDNTM	MINUTES (0-59)
(102)	258	BITSTRING	1		RESERVED FOR DUE INFO
(103)	259	BITSTRING	1	NJHPDGP3	GENERAL PURPOSE BYTE3
		1...		NJHPDG3G	"X'80'" LOG=NO SPECIFIED
		.1..		NJHPDG3M	"X'40'" 'EOJMSG WANTED' OPTION
		..1.		NJHPDG3Q	"X'20'" QUEUE FIX F. JOB CMPL MSG
		...1		NJHPDG3F	"X'10'" USER VALUE BY 'FROM=' OP.
		.... 1...		NJHPDG3J	"X'08'" QUEUE FIX F. JOB GEN MSG
		.... .1..		NJHPDG3C	"X'04'" QUEUE MSG TO COMM. QUEUE
		.... ..1.		NJHPDG3D	"X'02'" QUEUE MSG DOUBLE
		1		NJHPDG3R	"X'01'" *$$ JOB NORUN=IGN
(104)	260	CHARACTER	8	NJHPDIST	DISTRIBUTION CODE
(10C)	268	SIGNED	4	NJHPONUM	JOB NUMBER OF ORIG. JOB
(110)	272	CHARACTER	25	NJHPJSIN (0)	GCM SUBMIT-MSG INFO
(110)	272	CHARACTER	8	NJHPDAPL	JOB SUBMITTER'S APPLID
(118)	280	CHARACTER	8	NJHPDUID	JOB SUBMITTER'S USERID
(120)	288	CHARACTER	8	NJHPONOD	JOB SUBMITTER'S NODE NAME
(128)	296	BITSTRING	1	NJHPOQUL	JOB SUBMITTER'S SYSID/QUL
(129)	297	BITSTRING	2		RESERVED FOR FUTURE
(12B)	299	BITSTRING	1	NJHPDGP4	GENERAL PURPOSE BYTE 4
		1...		NJHPDG4A	"X'80'" .. JOB ECHO=ALL
		.1..		NJHPDG4R	"X'40'" .. JOB ECHO=REPLY
		..1.		NJHPDG4M	"X'20'" .. GCM R-MSG WANTED
		...1		NJHPDG4O	"X'10'" .. JOB ECHO=ONLY
		.... 1...		NJHPD4LM	"X'08'" .. LINE MODE STATE
		.... .1..		NJHPD4LI	"X'04'" .. LINE MODE IMM/IDM
		.... ..1.		NJHPD4PM	"X'02'" .. PAGE MODE STATE
		1		NJHPD4P8	"X'01'" .. PAGE MODE-B STATE
(12C)	300	CHARACTER	8	NJHPSID	SECURITY USERID
(134)	308	BITSTRING	8	NJHPSPW	SECURITY PASSWORD
(13C)	316	CHARACTER	8	NJHPSECN	SECURITY NODEID
(144)	324	BITSTRING	8	NJHPPRIV	USER'S PRIVATE INFORM'N
(14C)	332	CHARACTER	8	NJHPECHO	JOB ECHO USERID
(154)	340	CHARACTER	25	NJHPMRIN (0)	GCM R-MSG INFO
(154)	340	CHARACTER	8	NJHPMRAP	GCM R-MSG APPL. ID

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(15C)	348	CHARACTER	8	NJHPMRUS	GCM R-MSG USERID
(164)	356	CHARACTER	8	NJHPMRND	GCM R-MSG NODEID
(16C)	364	BITSTRING	1	NJHPMRSI	GCM R-MSG SYSTEMID
(16D)	365	CHARACTER	1	NJHPDGP5	GENERAL PURPOSE BYTE 5
		1...		NJHPDG5S	"X'80'" .. ENTRY NOTE SPOOL ACCESS PROTECTED
(16E)	366	CHARACTER	1		RESERVED FOR FUTURE USE
(16F)	367	BITSTRING	1	NJHPRQUL	SYSID WHERE JOB RECVED
(170)	368	SIGNED	4	NJHPMRON	GCM R-MSG ORIGINAL JOBNO
(174)	372	CHARACTER	32		RESERVED FOR FUTURE USE
(194)	404	SIGNED	4	NJHPEND (0)	END OF VSE/POWER SECTION
		1.11 11..		NJHPLLEN	"*-NJHP" LENGTH OF VSE/POWER SECTION
		EXPRESSION		NJHLLEN	"*-NJHDSECT" LENGTH OF ENTIRE BLOCK
		SECTION TYPE FLAGS			
		These flags are used by other operating systems which could be present within the network. The layouts of the sections used by these systems can be found in the appropriate operating system manual. Because a subsystem appears in this list of section type flags does not mean that IBM supports this subsystem as part of the network.			
		....		NTYPGEN	"B'00000000'" GENERAL SECTION
		1...		NTYPSUB	"B'10000000'" SUB SYSTEM SECTION
		1... 1..1		NTYPGDS	"B'10001001'" OUTPUT PROCESSING SECTION
		1... 1.1.		NTYPGJS	"B'10001010'" JOB SCHEDULING SECTION
		1... ...1		NTYPASP	"B'10000001'" ASP SUBSYSTEM SECTION
		1... ..1.		NTYPHASP	"B'10000010'" HASP SUBSYSTEM SECTION
		1... ..11		NTYPJES1	"B'10000011'" JES/RES SUBSYSTEM SECTION
		1... .1..		NTYPJES2	"B'10000100'" JES2 SUBSYSTEM SECTION
		1... .1.1		NTYPJES3	"B'10000101'" JES3 SUBSYSTEM SECTION
		1... .11.		NTYPPOWR	"B'10000110'" VSE/POWER SUBSYSTEM SECTION
		1... .111		NTYPVNET	"B'10000111'" VM/370 SUBSYSTEM SECTION
		11..		NTYPUSER	"B'11000000'" USER SECTION

Network Job Trailer Record (JTR)

Definition Macro: IPW$DNR JTR=YES

The Job Trailer record (JTR) is a control record which, together with the job header record, is used as the 'bounds' for jobs or output which are to be transmitted via the network. It contains several different sections and only those relevant to VSE/POWER are described in the following DSECT.

The layout of the first four bytes of every trailer record is identical. Bytes 0 and 1 are the length of the entire block. Individual records must not be greater than 256 bytes long, so if the total record is longer it must be segmented. Byte 2 is a flag byte that is zero. Byte 3 is the transmission sequence indicator and is used to indicate that a header has been segmented. The high order bit (X'1... ....') indicates that there are more parts to come, and the other bits are a sequence counter of the blocks for this record. For example, if the trailer had to be split into three parts then the sequence indicators in the three blocks would be as follows:- X'80', X'81', and X'02'.

The layout of the first four bytes of all sections is always identical. The section flags are to be found in the description of the job header record (refer to “Network Job Header Record (JHR)” on page 559).

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
					Network Job Trailer Record (JTR)
(0)	0	STRUCTURE		NJTDSECT	
	BLOCK CONTROL INFORMATION				
(0)	0	ADDRESS	2	NJTLEN	LENGTH OF ENTIRE BLOCK
(2)	2	BITSTRING	1	NJTFLAGS	FLAGS
(3)	3	BITSTRING		NJTSEQ	TRANSMISSION SEQUENCE INDICATOR
		.... .1..		NJTLBCI	"*-NJTDSECT" LENGTH OF BLOCK CONTROL INFORMAT
	GENERAL SECTION				
(4)	4	SIGNED	4	NJTG (0)	START OF GENERAL SECTION
(4)	4	ADDRESS	2	NJTGLEN	LENGTH OF GENERAL SECTION
(6)	6	BITSTRING	2	NJTGFLGS (0)	SECTION TYPE FLAGS
(6)	6	ADDRESS	1	NJTGTYPE	ID FOR GENERAL SECTION
(7)	7	ADDRESS	1	NJTGMOD	MODIFIER
		....		NJTG$MOD	"B'00000000'" .. VALUE OF MODIFIER
(8)	8	BITSTRING	1	NJTGFLG1	FLAGS
(9)	9	CHARACTER	1	NJTGXCLS	ACTUAL EXECUTION CLASS
(A)	10	BITSTRING	2		RESERVED
(C)	12	SIGNED	8	NJTGSTRT	EXECUTION START TIME/DATE
(14)	20	SIGNED	8	NJTGSTOP	EXECUTION STOP TIME/DATE
(1C)	28	SIGNED	4	NJTGACPU	ACTUAL CPU TIME
(20)	32	SIGNED	4	NJTGALIN	ACTUAL OUTPUT LINES
(24)	36	SIGNED	4	NJTGACRD	ACTUAL OUTPUT CARDS
(28)	40	SIGNED	4	NJTGEXCP	EXCP COUNT
(2C)	44	ADDRESS	1	NJTGIXPR	INITIAL XEQ SELECTION PRIORITY
(2D)	45	ADDRESS	1	NJTGAXPR	ACTUAL XEQ SELECTION PRIORITY
(2E)	46	ADDRESS	1	NJTGIOPR	INITIAL OUTPUT SELECTION PRIORITY
(2F)	47	ADDRESS	1	NJTGAOPR	ACTUAL OUTPUT SELECTION PRIORITY
(30)	48	SIGNED	4	NJTGEND (0)	END OF GENERAL SECTION
		..1. 11..		NJTGLLEN	"*-NJTG" LENGTH OF GENERAL SECTION
		..11		NJTLLEN	"*-NJTDSECT" LENGTH OF ENTIRE BLOCK

Network Presentation Work Area

Definition Macro: IPW$DWP

This work area is used by the receiver. Presentation services is responsible for taking a transmission block, decompressing it and passing individual records to the receiver.

Offset Hex	Type	Len	Name (Dim)	Description
(0)	STRUCTURE	264	NPWA	RECORD PRESENTATION WORKAREA
(0)	CHARACTER	128	NPWDYNA	PLS DYNAMIC DATA AREA
(80)	CHARACTER	4	NPWDSD	LITTLE STORAGE DESCRIPTOR
(84)	BITSTRING	1	NPWRC	RETURN CODE PRESENTATION SERVICE
(85)	CHARACTER	3	*	RESERVED
(88)	ADDRESS	4	NPWBUPRO	POINTER TO PROCESS BUFFER
(8C)	ADDRESS	4	NPWRCDPT	ADDRESS OF PHYSICAL RECORD IN NORMAL OR ALTERNATE PRES. BUFFER
(90)	ADDRESS	4	NPWHDRPT	PTR TO HEAD ACCUMULATED RCD
(94)	ADDRESS	4	NPWPRBUF	PTR TO PRESENTATION BUFFER
(98)	ADDRESS	4	NPWALBUF	PTR TO ALTERNATE PRES BUFFER
(9C)	UNSIGNED	2	NPWTRCL	ACCUMULATED LENGTH OF SEGMENTS
(9E)	UNSIGNED	2	NPWFSGTL	TOTAL LENGTH OF SPANNED RECORD AS INDICATED IN FIRST SEGMENT
(A0)	UNSIGNED	2	NPWPRBLN	LENGTH OFF PRESENTATION BUFFER
(A2)	UNSIGNED	2	NPWALBLN	LENGTH ALTERNATE PRES BUFFER
(A4)	SIGNED	2	NPWALUSE	USE COUNT ALTERNATE BUFFER
(A6)	CHARACTER	1	NPWLPREQ	RESERVED
	1...		NPWSPAN	REQ ACCUMULATE RECORD SEGMENTS
	.1..		NPWNXTSG	REQ ACCUMULATE HEADER SEGMENTS
	..11 1111		*	RESERVED
(A7)	CHARACTER	1	*	RESERVED
(A8)	CHARACTER	56	NPWFSVE	FUNCTION SAVE AREA IMPLICIT LENGTH DEFINITION
(E0)	CHARACTER	40	NPWDKA	DECOMPRESSION WORK AREA IMPLICIT LENGTH DEFINITION
(108)	CHARACTER		NPWAEND	END OF PRESENTATION WORK AREA

Network Receiver Work Area

Definition Macro: IPW$DWG

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
				Network Receiver Workarea	
(0)	0	STRUCTURE	1248	NRWA	RECEIVER WORKAREA
(0)	0	CHARACTER	80	NRDYNA	PLS DYNAMIC DATA AREA
(50)	80	CHARACTER	1168	NRWAP1E	2ND PART OF WORKAREA

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
				(2nd part of receiver workarea)	
(0)	0	STRUCTURE	1168	NRWAP2	RCV WORKAREA PART 2
(0)	0	CHARACTER	256	NR2DYNA	PLS DYN DATA AREA FOR $$NR2
(100)	256	CHARACTER	56	NR2SFSV	
SAVE AREA OF $$NR2					
(138)	312	CHARACTER	16	NRDSD	NRWA STORAGE DESCRIPTOR
(148)	328	ADDRESS	4	NRTCB	POINTER TO RECEIVER TCB
(14C)	332	UNSIGNED	1	NRSECTRC	RETURN CODE FIND POWER SECT RTN
(14D)	333	UNSIGNED	1	NRSTATUE	STATUS BYTE USER EXIT RTN
		1...		NRUSREXT	USER EXIT RTN LOADED
		.1..		NRUSBTRN	USER REQ NO BLANK TRUNCATION@DA43344
		..11 1111		*	RESERVED
(14E)	334	BITSTRING	1	*	RESERVED
(14F)	335	UNSIGNED	1	NRDSRC	RETURN CODE DS-HEADER SCAN
(150)	336	UNSIGNED	1	NRALLCRC	RETURN CODE CB-ALLOCATION RTN
(151)	337	BITSTRING	1	NRRQALLC	ALLOCATION REQUEST BYTE
		1...		NRRQQREC	REQUEST FOR QUEUE RECORD
		.1..		NRRQDBLK	REQUEST FOR DBLK AREA
		..1.		NRRQDSCB	REQUEST FOR DSCB ENTRY
		...1		NRRQINDS	REQUEST FOR DSCB INITIALIZATION
		.... 1...		NRRQQRNO	REQ FOR NO JOB NUMBER
		.... .1..		NRRQQRPW	REQ TO USE POWER DEFAULTS
		11		*	RESERVED
(152)	338	UNSIGNED	2	NRUSRRC	RETURN CODE USER EXIT RTN
(152)	338	UNSIGNED	1	NRUSRRC2	RETURN CODE 2 USER EXIT RTN
		1...		NRUSRCBT	USER REQ NO BLANK TRUNCATION@DA43344
		.111 1111		*	RESERVED
(153)	339	UNSIGNED	1	NRUSRRC1	RETURN CODE 1 USER EXIT RTN
(154)	340	SIGNED	2	NRREASON	TERMINATION REASON CODE
(156)	342	SIGNED	2	NR#SPREQ	NBR OF OUTPUT SPOOL REQUESTS
(158)	344	ADDRESS	4	NRJHDR	PTR TO JOB HEADER
(15C)	348	CHARACTER	8	NRTCBRW (2)	RECORD CONTROL WORD LIST
(15C)	348	BITSTRING	1	NRTCBCC	COMMAND CODE
(15D)	349	ADDRESS	3	NRTCBRV	RECORD ADDRESS
(160)	352	BITSTRING	1	NRTCBGP	GENERAL PURPOSE BYTE 1 (SEE TCB)
		1...		*	RESERVED
		.1..		NRGPBSR	SEGMENTED/EXTENDED RECORD
		..1.		NRGPBDE	END OF 3540 RECORD
		...1		NRGPBEB	END OF BLOCK

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
		.... 1...		*	RESERVED
		.... .1..		NRGPBED	END OF DATA
		.... ..1.		NRGPBRD	3540 DATA RECORD
		1		NRGPBDR	LINE PRINT/CARD MOVE
(161)	353	BITSTRING	1	NRTCBG2	GENERAL PURPOSE BYTE 2
		1...		NRTCBJHR	JOB HEADER RECORD
		.1..		NRTCBJTR	JOB TRAILER RECORD
		..1.		NRTCBDHR	DATA SET HEADER RECORD
		...1		NRTCBPRI	PAGE RECORD IDENTIFIER
		.... 11..		*	UNUSED BY RECEIVER
		.... ..1.		NRTCBASA	ASA DATA RECORD INDICATOR
		1		*	UNUSED BY RECEIVER
(162)	354	SIGNED	2	NRTCBRL	RECORD LENGTH
(16C)	364	ADDRESS	4	NRDSINIT	PTR TO INITIAL DSCB ENTRY
(170)	368	ADDRESS	4	NRDSNEW	PTR TO NEW DSCB ENTRY
(174)	372	ADDRESS	4	NRDSSRCH	PTR TO SCAN ALONG DSCB CHAINS
(178)	376	ADDRESS	4	NRDSCUR	PTR TO CURRENT DSCB IN ACT CHAIN
(17C)	380	ADDRESS	4	NRDSACT	PTR TO ACTIVE DISTR CHAIN
(180)	384	ADDRESS	4	NRDSSUSP	PTR TO SUSPENDED DISTR CHAIN
(184)	388	ADDRESS	4	NRPRBUF	PTR TO PRESENTATION BUFFER
(188)	392	ADDRESS	4	NRLR	PTR TO LOGICAL RECORD
(18C)	396	ADDRESS	4	NRUSRRCD	PTR TO USER RECORD
(190)	400	SIGNED	2	NRUSRLEN	LENGTH USER RECORD
(192)	402	SIGNED	2	NRUSRBLN	LENGTH INTERMEDIATE DATA BUFFER
(194)	404	ADDRESS	4	NRUSRBUF	PTR TO INTERMEDIATE DATA BUFFER
(198)	408	ADDRESS	4	NRRCDSAV	PTR TO SAVED CURRENT RECORD
(19C)	412	ADDRESS	4	*	UNUSED
(1A0)	416	BITSTRING	1	NRPRSRCB	SRCB CHAR PRECEDING RECORD
(1A1)	417	BITSTRING	1	NRSTAT1	STATUS BYTE 1
		1...		NRST1JJ	SEARCH FOR $$ JOB
		.1..		NRST1JF	ONCE A $$ JOB FOUND
		..1.		NRST1JC	CONTINUATION TO PROC
		...1		NRST1JW	WRITE JOB HDR RECORD
		.... 1...		NRST1JE	ERROR FOUND
		.... .1..		NRST1JR	RCVED $$ JOB
		.... ..1.		NRST1PD	PROPAGATE DATASET HDR
		1		NRST1DD	DON'T SPOOL DSHR
(1A2)	418	BITSTRING	1	*	FREE FOR FUTURE USE
(1A3)	419	BITSTRING	1	*	FREE FOR FUTURE USE
(1A4)	420	SIGNED	4	NRSAVRD	R13-SAVE AREA IN $$NR2
(1A8)	424	BITSTRING	1	NRORGQ	ORIGINATOR NODE QUALIFIER, USED FOR NOTIFICATION.
(1A9)	425	CHARACTER	22	*	RESERVED
(1BF)	447	CHARACTER	BITSTRING	NRSTAT2	STATUS BYTE 2
		1...		NRST2RR	DSHR RCCS RECEIVED
	FUNCTION WORK AREAS				
(1C0)	448	CHARACTER	128	NRACOUNT	RECEIVER ACCOUNT AREA
(240)	576	CHARACTER	264	NRPWA	PRESENTATION WORK AREA
(348)	840	CHARACTER	328	NRCWA	COMPOSER WORK AREA
(490)	1168	CHARACTER		NRWAEND	START OF PRESENTATION BUFFER

Network Transmitter Work Area

Definition Macro: IPW$DWG

Offset Hex	Type	Len	Name (Dim)	Description
(0)	STRUCTURE	1267	NTWA	TRANSMITTER WORKAREA
(0)	CHARACTER	128	NTDYNA	PLS DYNAMIC DATA AREA
(80)	CHARACTER	16	NTDSD	NTWA STORAGE DESCRIPTOR
(90)	ADDRESS	4	NTTCB	PTR TO TASK CONTROL BLOCK
(94)	ADDRESS	4	NTANCWA	PTR TO COMPOSER WORK AREA
(98)	ADDRESS	4	NTANMR	PTR TO CURRENT NODAL MESSAGE RECORD
(9C)	ADDRESS	4	NTTCBRV	PTR TO POWER RECORD
(A0)	ADDRESS	4	NTSCTPTR	POINTER TO HEADER SECTION
(A4)	ADDRESS	4	NTCNTEND	POINTER TO END OF CONTROL RECORD
(A8)	ADDRESS	4	NTLPTR	LOOP CONTROL POINTER IN ABTERM
(AC)	ADDRESS	4	NTNCBPTR	POINTER TO NCB
(B0)	BITSTRING	1	NTIND	INDICATOR BYTE
	1...		NTABRT	ABORT INDICATOR
	.1..		NTNOJB	NO JOB AVAILABLE
	..1.		NTNFY	SEND NTFY-MSG. INDICATOR
	...1		NTSS	INTERN. STOPSTATE INDICATOR
	.... 1...		NTEMERGE	INTERN. LINE ERROR/SIGNOFF IND.
	.... .1..		NTJOBRUN	LOOP CONTROL VARIABLE .
	11		*	RESERVED
(B1)	BITSTRING	1	NTSAVST	SAVED STOP STATE ??
(B2)	BITSTRING	1	NTREASON	REASON CODE FOR MESSAGES
(B3)	UNSIGNED	1	NTSECTRC	RETURN CODE FIND POWER SECTION ROUTINE
(B4)	CHARACTER	8	NTUSID	ORIGINATOR NODE USER ID
(BC)	CHARACTER	8	NTORGN	ORIGINATOR NODE NAME, USED FOR NOTIFICATION
(C4)	BITSTRING	1	NTORGQ	ORIGINATOR NODE QUALIFIER, USED FOR NOTIFICATION.
(C5)	CHARACTER	23	*	UNUSED
(DC)	CHARACTER	20	NTEXDS	XMTEXIT PARAMETER LIST
(DC)	ADDRESS	4	NTEXRV	RECORD ADDRESS
(E0)	UNSIGNED	4	NTEXRL	RECORD LENGTH
(E4)	BITSTRING	1	NTEXCC	OPERATION CODE
(E5)	BITSTRING	1	NTEXRT	RECORD TYPE
(E6)	BITSTRING	1	NTEXDT	DATA/CONTROL REC TYPE
(E7)	BITSTRING	1	NTEXOT	DATA STREAM INDICATOR
(E8)	ADDRESS	4	NTEXWA	ADDR. OF EXIT WORK AREA
(EC)	BITSTRING	1	NTEXRC	RETURN CODE FROM EXIT
(ED)	CHARACTER	3	NTEXDU	RESERVED FOR FUTURE
(F0)	ADDRESS	4	NTUEXWA	PTR TO XMTEXIT WORK AREA
(F4)	UNSIGNED	4	NTUEXWAL	LENGTH OF XMTEXIT W.A.
(F8)	CHARACTER	8	NTRCWSA	SAVE AREA FOR TCBRW
(100)	ADDRESS	4	NTSAVD	SAVE FIELD FOR R13
(104)	ADDRESS	4	NTSVJTR	WORK AREA FOR DEF. JTR
(108)	BITSTRING	1	NTHDLFLG	XMTEXIT HANDLER FLAG
(109)	BITSTRING	1	NTEHDLRC	RECORD DELETE INDICATOR
(10A)	CHARACTER	2	*	FULLWORD ALIGNMENT
(10C)	CHARACTER	12	NTNDHC	DATASET HEADER BLOCK.. CNTL INFO + CHANGE SECTION
(118)	CHARACTER	8	NTTCBRW	SAVE AREA FOR TCBRW

Offset Hex	Type	Len	Name (Dim)	Description
(120)	CHAR-ACTER	128	NTNACT	STORAGE FOR ACCOUNT RECORD
(1A0)	CHAR-ACTER	328	NTNCWA	STORAGE FOR COMPOSER WORK AREA
(2E8)	CHAR-ACTER	259	NCSGAR	STORAGE FOR COMPOSER SEGMENT AREA: TP BUFFER-SIZE + SNA-RID-LENGTH
(3EB)	CHAR-ACTER	264	NCKOUT	STORAGE FOR COMPRESSION OUTP. AREA
(4F3)	CHAR-ACTER		NTEND	

Network Transmitter Exit Parameter List

Definition Macro: IPW$DTX

This macro is used to produce a DSECT for the Transmitter Exit Parameter List. The format is as follows:

Bytes Hex.	Label of Field	Description/Function
00	TEXDS	Start of DSECT
00-03	TEXRV	Record address of statement passed
04-07	TEXRL	Length of statement passed
08	TEXCC	Operation code
09	TEXRT	Record type
	TERNCD	X'00' - Normal data or control record
	TERJHR	X'80' - Job header record
	TERJTR	X'40' - Job trailer record
	TERDSHR	X'20' - Data set header record
0A	TEXDT	Type of data stream
	TEDJRNC	X'00' - not defined
	TEDCPDS	X'10' - CPDS record
	TEDASA	X'02' - ASA record
	TEDLPCM	X'01' - Line print/card move record
0B	TEXOT	Various information
	TEOLST	X'80' - Output from list queue
	TEOPUN	X'40' - Output from punch queue
	TEOJOB	X'20' - Job data
0C-0F	TEXWA	Pointer to exit work area
10	TEXRC	Return codes
	TEROK	X'00' - Process record
	TERDEL	X'04' - Delete this record
	TERINS	X'08' - Insert new record
	TERFLS	X'10' - Flush queue entry
	TERMOD	X'14' - Process modified network control record
	TERFLSH	X'18' - Flush queue entry with HOLD
11-13		Reserved for future use

Nodal Message Record (NMR)

Definition Macro: IPW$DNR NMR=YES

The Nodal message record (NMR) is the record format used to transmit all messages and commands throughout the network.

Bytes Hex.	Label of Field	Description/Function
00	NMRFLAG	Flag byte
	NMRFLAGC	X'80' - NMRMSG contains a command
	NMRFLAGW	X'40' - NMROUT has VSE/POWER remote number
	NMRFLAGT	X'20' - NMROUT has a ICCF/CMS userid
	NMRFLAGU	X'10' - NMROUT has UCMID information
		* The next four flag settings are
		* not used by VSE/POWER.
	NMRFLAGR	X'08' - Console is only remote authorized
	NMRFLAGJ	X'04' - Console is not job authorized
	NMRFLAGD	X'02' - Console is not device authorized
	NMRFLAGS	X'01' - Console is not system authorized
	NMRFLAGN	X'0F' - Non-trusted user
01	NMRLEVEL	Importance level (high 4 bits)
	NMRPRIO	Output priority (low 4 bits)
02	NMRTYPE	Type byte
	NMRTYPEO	X'80' - Operator Authority
	NMRTYPEX	X'60' - Reserved
	NMRTYPE5	X'10' - Message contains Application ID
	NMRTYPE4	X'08' - Message text contains control information
	NMRTYPET	X'04' - Message text only in NMRMSG
	NMRTYPEF	X'02' - Formatted command in NMRMSG
	NMRTYPED	X'01' - 'DOM' (not supported)
03	NMRML	Message length
04-0C	NMRTO	Target node
04-0B	NMRTONOD	Target node name
0C	NMRTOQUL	Target node qualifier
0D-14	NMROUT	Local output information
15-1D	NMRFM	Originator node
15-1C	NMRFMNOD	Originator node name
1D	NMRFMQUL	Originator node qualifier
1E-A2	NMRMSG	Message

Node Control Block (NCB)

Definition Macro: IPW$DNC

The Node Control Block (NCB) is used in the PNET environment to control all actions concerned with a connection to another node in the network. It is created whenever a PSTART PNET,nodeid.... is given by the operator and is deleted after the connection is terminated. The address of the first NCB is found from the PNCB, label PNCBNCB.

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
NODE CONTROL BLOCK					
A NODE CONTROL BLOCK (NCB) IS CREATED FOR EACH NODE TO BE STARTED. ALL NCB ARE CHAINED TOGETHER. THE POINTER TO THE FIRST NCB IS CONTAINED IN THE PNCB NETWORK DEFINITION TABLE HEADER					
(0)	0	STRUC-TURE	0	NCBDS	, DEFINE DUMMY SECTION
(0)	0	CHAR-ACTER	16	NCBSD	SECTION DESCRIPTOR
(10)	16	CHAR-ACTER	8	NCBNAME	NODE NAME
(18)	24	ADDRESS	4	NCBNEXT	ADDRESS OF NEXT NCB IN CHAIN
(1C)	28	ADDRESS	4		RESERVED FOR LOCKWORD
(20)	32	BITSTRING	1	NCBTYP	NODE TYPE
		1...		NCBSNA	"X'80'" .. SNA NODE, IF OFF = BSC NODE
		.1..		NCBCTCA	"X'40'" .. CTC ADAPTER
		..1.		NCBTCP	"X'20'" .. TCP/IP NODE - FOR THIS .. TYPE 'NCBCTCA' IS SET TOO
		...1		NCBSSL	"X'10'" .. TCP/IP NODE WITH SSL TYPE 'NCBCTCA' IS SET TOO TYPE 'NCBTCP ' IS NOT SET
		.... 1...		NCBRSTR	"X'08'" .. ON IF NO RESTART AFTER TIME-OUTS
(21)	33	BITSTRING	2	NCBACTB (0)	ACTION BYTES
(21)	33	BITSTRING	1	NCBACT1	ACTION BYTE 1
		1...		NCBDTCH	"X'80'" .. TASK DETACH REQUESTED
		.1..		NCBTCRQ	"X'40'" .. TASK CREATION REQUESTED
		..1.		NCBLNSR	"X'20'" .. LINE STOP REQUESTED
		...1		NCBFREE	"X'10'" .. NCB FREE REQUESTED (SNA)
		.... 1...		NCBINIT	"X'08'" .. LINE START REQUESTED
		.... .1..		NCBIPEND	"X'04'" .. INIT PENDING CTCA ONLY
		.... ..1.		NCBREACT	"X'02'" .. RESTART ACTIVITY
		1		NCBTPEND	"X'01'" .. INIT PENDING TCP/IP ONLY
(22)	34	BITSTRING	1	NCBACT2	ACTION BYTE 2
		1...		NCBSGNR	"X'80'" .. SIGNON PROCEDURE REQUESTED
		.1..		NCBSOFR	"X'40'" .. SIGNOFF PROCESSING RQSTD
		..1.		NCBLCLS	"X'20'" .. LINE CLOSE PROCESSING REQUEST.
		...1		NCBSOFR	X'10' .. NOT USED
		.... 1...		NCBSEND	"X'08'" .. BUFFER READY TO SEND
		.... .1..		NCBSOFR	X'04' .. NOT USED
		.... ..1.		NCBRCVR	"X'02'" .. RECEIVE REQUESTED (SNA)
		1			X'01' .. RESERVED
(23)	35	BITSTRING	1	NCBPROC	PROCESS BYTE
		1...		NCBPSGN	"X'80'" .. SIGNON IN PROCESS
		.1..		NCBPRST	"X'40'" .. AUTOMATIC RESTART IN PROCESS
		...1			X'20' .. NOT USED
		...1		NCBTRCE	"X'10'" .. LINE TRACE MODE
		.... 1...			X'08' .. RESERVED
		.... .1..		NCBDLAY	"X'04'" .. DELAYED RESPONSE IN PROCESS
(24)	36	BITSTRING	1	NCBFLG1	STATUS BYTE 1 (NODE)
		1...		NCBSGNOC	"X'80'" .. SIGNON COMPLETED
		.1..		NCBSGOP1	"X'40'" .. PART1 OF SIGN-ON PROCESS FINISHED
		..1.		NCBSGNOS	"X'20'" .. SIGNON RECORD OUT OF SEQUENCE
		.... 1...		NCBSGNR1	"X'08'" .. SIGNON PROCEDURE 1
		.... .1..		NCBSGNR2	"X'04'" .. SIGNON PROCEDURE 2
		.... ..1.		NCBSGOP2	"X'02'" .. PART2 OF SIGN-ON FINISHD
(25)	37	BITSTRING	1	NCBFLG2	STATUS BYTE 2
		.... 1...		NCBNSCLN	"X'08'" .. NSEXIT DRIVEN (CLEANUP)
		.... .1..		NCBCTCR	"X'04'" CTC RESTART FLAG
		.... ..1.		NCBWRDI	"X'02'" RESTART BECAUSE WRONG MEMBER DIALED
		1		NCBWACT	"X'01'" WRITE STATISTIC AND ACCOUNT RECORD .. NOT USED
(26)	38	BITSTRING	1	NCBLTTC	.. NOT USED
(27)	39	BITSTRING	1	NCBTTC	TERMINATION CODE
		1...		NCBTTCV	"X'80'" .. STOP DUE TO VTAM ABEND (SNA)
		.1..		NCBTTCL	"X'40'" .. STOP DUE TO LINE/SESSION ERROR

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
		..1.		NCBTTCO	"X'20'" .. STOP DUE TO SIGNOFF RCVD (BSC)
		...1		NCBTTCS	"X'10'" .. STOP IMMEDIATE
		.... 1...		NCBTTCE	"X'08'" .. STOP AT EOJ
		.... .1..		NCBTTCF	"X'04'" .. STOP IMMEDIATE,FORCE
		.... ..1.		NCBTTCC	"X'02'" .. STOP DUE CTCA COLISION
(28)	40	BITSTRING	1	NCBTTCDE	ERROR SUB-CODE
(29)	41	BITSTRING	1	NCBTTSTQ	STOP QUALIFIER FOR NCBTTCS/NCBTTCE
		1...		NCBPHALT	"X'80'" .. PNET HALTS NODE DUE TO ERROR
		.1..		NCBVHALT	"X'40'" .. VTAM REQUESTED NODE STOP AT EOJ
		..1.		NCBRHALT	"X'20'" .. REMOTE VTAM NODE REQUESTED EOJ
		...1		NCBOHALT	"X'10'" .. OPERATOR STOP PSTOP/PEND
		.... 1...		NCBSHALT	"X'08'" .. HALT - SEVERE LOGIC ERROR
(2A)	42	ADDRESS	2	NCBCNRV	CURR. NUMBER OF RECEIVERS ACTIVE
(2C)	44	ADDRESS	2	NCBCNTR	CURR. NUMBER OF TRANSMITTERS ACTIVE
		.... .111		NCBMNRV	"7" .. MAX NUMBER OF RECEIVERS
		.... .111		NCBMNTR	"7" .. MAX NUMBER OF TRANSMITTERS
(2E)	46	ADDRESS	2		PADDING BYTES

THE FOLLOWING TABLE DEFINES THE TASK ENTRIES FOR THE
COMMAND / MESSAGE TRANSMITTER RECEIVER (CONSOLE TR/RV)
ALL TABLES MUST BE ADJACENT .

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(30)	48	BITSTRING	8	NCBCONST	CONSOLE TRANSMITTER TASK
(38)	56	BITSTRING	8	NCBCONSR	CONSOLE RECEIVER TASK

THE FOLLOWING TABLES DEFINE THE TASK ENTRIES FOR THE
TRANSMITTERS.

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(40)	64	BITSTRING	56	NCBJTTB	JOB TRANSMITTER TABLE
(78)	120	BITSTRING	1	NCBOTTB	OUTPUT TRANSMITTER TABLE

THE FOLLOWING TWO TABLES DEFINE THE VARIOUS TASK ENTRIES
FOR THE RECEIVERS.

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(B0)	176	BITSTRING	56	NCBJRTB	JOB RECEIVER TABLE
(E8)	232	BITSTRING	56	NCBORTB	OUTPUT RECEIVER TABLE
(120)	288	ADDRESS	4	NCBMSGA	POINTER TO FIRST MSG/CMND IN QUEUE
(124)	292	ADDRESS	4	NCBMSGT	MSG/CMND QUEUE TAIL POINTER
(128)	296	ADDRESS	4	NCBNBFRQ	PTR TO SIGNON RECORD RECEIVED

BUFFER CONTROL FIELDS

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(12C)	300	ADDRESS	4	NCBIFRE	ADDR OF FREE INPUT BUFFER QUEUE
(130)	304	ADDRESS	4	NCBOTBS	ADDR TO-BE-SENT QUEUE (NO PRIORITY)
(134)	308	ADDRESS	4	NCBOBTL	TAIL PTR TO-BE-SENT QUEUE (NO PRI.)
(138)	312	ADDRESS	4	NCBOTBP	HEAD PTR PRIORITY BUFFER Q
(13C)	316	ADDRESS	4	NCBOBTP	TAIL PTR PRIORITY OUTPUT Q
(140)	320	ADDRESS	4	NCBLBFI	LINE DRIVER BUFFER (BSC/CTC)
(144)	324	ADDRESS	4	NCBLBFO	LINE DRIVER BUFFER (BSC/CTC)
(148)	328	ADDRESS	2	NCBBFSZ	PNET BUFFER SIZE
(14A)	330	ADDRESS	1	NCBMNIB	MAX. NO OF INPUT BUFFERS
(14B)	331	ADDRESS	1	NCBMNJB	MAX. NO OF JOB/OUT XMIT BUFFERS
(14C)	332	ADDRESS	2	NCBNIBU	NUMBER OF ACQUIRED INPUT BUFFERS
(14E)	334	ADDRESS	2		PADDING BYTES

I/O AND SEND/RECEIVE MANAGER FIELDS

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(150)	336	ADDRESS	4	NCBIBUF	ADDR OF NON-LINE I-BFR (BSC/CTC)
(154)	340	ADDRESS	4	NCBOBUF	ADDR OF NON-LINE O-BFR (BSC/CTC)
(158)	344	ADDRESS	4	NCBCBFI	ADDR OF BUFFER FOR ACTUAL RECEIVE
(15C)	348	ADDRESS	4	NCBCBFO	ADDR OF BUFFER FOR ACTUAL SEND
(160)	352	BITSTRING	2	NCBRFCS	REMOTE HELD/RELEASED STREAM STATUS
(162)	354	BITSTRING	2	NCBTFCS	NEW HELD/RELEASED STREAM STATUS
		.1..		NCBFCSWB	"X'40'" WAIT-A-BIT .. (SUSPENDED ALL STREAMS)

QUEUE HEADER FOR 'PARKED' BUFFERS

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(164)	356	ADDRESS	4	NCBPBFRQ	HEADER OF SUSPENDED BUFFERS
(168)	360	ADDRESS	4	NCBNDTEN	ADDR OF NDT ENTRY
(16C)	364	ADDRESS	4		RESERVED
(170)	368	ADDRESS	4		RESERVED

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(174)	372	ADDRESS	4		RESERVED *

START OF BSC/CTC SECTION
LINE AND SESSION RELATED FIELDS
THE FIELDS CONTAIN THE INFORMATION ON THE STATUS OF THE LINK
BEING USED FOR SEND/RECEIVE, I.E. THE SNA SESSION RESPECTIVELY
THE BSC LINE. AS A NODE CAN BE ONE OR THE OTHER THIS FIELD IS
AN OVERLAY.

(178)	376	DBL WORD	8	NCBLCB (0)	START OF CONTROL BLOCK OVERLAY

BSC CTC LINE CONTROL INFORMATION

(178)	376	ADDRESS	4	NCBPUBA	ADDR OF RELATED PUB ENTRY FOR CUU
(17C)	380	BITSTRING	1	NCBEBCB	EXPECTED BCB (BSC)
(17D)	381	BITSTRING	1	NCBTBCB	TRANSMITTED BCB (BSC)
(17E)	382	BITSTRING	1	NCBREQF	REQUEST FIELD FOR I/O MANAGER
(17F)	383	BITSTRING	1	NCBLREQ	LAST SENT REQUEST
(180)	384	BITSTRING	1	NCBNNAK	LAST SENT NON NAK REQUEST
(181)	385	BITSTRING	1	NCBLDRQ	LAST REQUEST BY LINE-DRIVER
(182)	386	BITSTRING	1	NCBIOF1	FLAG BYTE FOR I/O MGR/LDR COMMUN.
		1...		NCBIOBF	"X'80'" .. BUFFER TO ACKNOWLEDGE
		.1..		NCBTERM	"X'40'" .. STOP I/O FOR LINE
		..1.		NCBBCBL	"X'20'" .. BCB NOT TO BE UPDATED
(183)	387	BITSTRING	1	NCBTOCT	TIMEOUT COUNT FOR SWITCHED LINES (PRE-SIGNON)
(184)	388	BITSTRING	1	NCBRTRY	RETRY COUNT FOR UNIT CHECK (MAX 30)
(185)	389	BITSTRING	1	NCBRCNT	RETRY COUNT
(186)	390	BITSTRING	1	NCBTIMC	TIME OUT COUNT
(187)	391	BITSTRING	1	NCBLFB1	FLAG BYTE 1
		1...		NCBF1BY	"X'80'" .. LINE BUSY
		.1..		NCBF1SEC	"X'40'" .. REMOTE NODE IS SECONDARY
		..1.		NCBF1CON	"X'20'" .. CONTENTION DETECTED
		...1		NCBF1CUE	"X'10'" .. CONTENTION DETECTED BY UNIT EXCEPTION HANDLER.
		.... 1...		NCBF1FPR	"X'08'" .. FORCE PRIMARY
(188)	392	BITSTRING	1	NCBLFB2	FLAG BYTE 2
		1...		NCBF2IN	"X'80'" .. LINE INITIALIZED
(189)	393	BITSTRING	2	(0)	ALIGN
(189)	393	BITSTRING	1		UNUSED
(18A)	394	BITSTRING	1	NCBINTC	CTCT INIT NO OF 1.5 MIN TIME .. OUTS
(18B)	395	BITSTRING	1	NCBCTCCB	CTC COMMAND BYTE

THE FOLLOWING FIELDS REPRESENT THE LINE BLOCK ENTRY
AS SPECIFIED IN THE PLINE MACRO FOR THE ASSOCIATED CUU.

(18C)	396	SIGNED	4	NCBPLINE (0)	

VSE/POWER - PLINE - 5686-066-03

(18C)	396	ADDRESS	2	NCBLPU	PHYSICAL UNIT ADDRESS
(18E)	398	ADDRESS	2	NCBTLIM	TIME OUT LIMIT(SECONDS) LINE FEATURES
(190)	400	ADDRESS	1	NCBFEA1	DUAL MODE
(191)	401	ADDRESS	1	NCBLDM	
		.... .11.		NCBL1	"*-NCBLPU" LENGTH WITHOUT PASSWORD
(192)	402	CHAR-ACTER	8	NCBLPW	LINE PASSWORD
(19A)	410	BITSTRING	2		UNUSED
		...1		NCBL	"*-NCBLPU" LENGTH OF LINE TAB ENTRY
(19A)	410		0	NCBNXT	"*" NEXT LINE TAB ENTRY

CCB AND CCW'S AND TEMPORARY WORKAREA

(19C)	412	BITSTRING	4		ALIGN
(1A0)	416	BITSTRING	24	NCBCCB	NJE CCB
(1B8)	440	DBL WORD	8	NCBCCW (5)	CHANNEL PROGRAM
(1E0)	480	SIGNED	4	NCBLCCW	ADDRESS OF LAST EXECUTED CCW
(1E4)	484	SIGNED	4	NCBDISP	DISPL BETWEEN REAL - VIRTUAL OF NCB
(1E8)	488	BITSTRING	2	NCBSENS (0)	SENSE BYTES
(1E8)	488	BITSTRING	1	NCBSNS1	FIRST SENSE BYTE
(1E9)	489	BITSTRING	1	NCBSNS2	SECOND SENSE BYTE

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(1EA)	490	BITSTRING	2	NCBSNSLA	SENSE BYTES OF LAST I/O
(1EA)	490		0	NCBBSCE	"*" END OF BSC PART
SNA SESSION CONTROL INFORMATION					
(178)	376	CHAR-ACTER	8	NCBAPLID	APPL-ID OF REMOTE NODE
(180)	384	ADDRESS	4	NCBSSCB	ADDRESS OF SSCB
(184)	388	ADDRESS	4	NCBSRQE	ADDRESS OF SRQE
(188)	392	ADDRESS	4	NCBCTCB1	ADDR. CONNECT SESSION TCB LOCAL
(18C)	396	ADDRESS	4	NCBCTCB2	ADDR. CONNECT SESSION TCB REMOTE
(190)	400	ADDRESS	4	NCBDTCB	ADDRESS OF DISCONNECT SESSION TCB
(194)	404	ADDRESS	4	NCBDECB	ECB OF DISCONNECT TASK
(198)	408	ADDRESS	4	NCBIFREX	ADDR 'RECEIVE-AHEAD' IPT BFR QUEUE
(19C)	412	ADDRESS	4	NCBOTBSX	ADDR 'SEND-AHEAD' OUTPUR BFR QUEUE
(1A0)	416	ADDRESS	4	NCBOBTLX	TAIL PTR 'SEND-AHEAD' OUTPUT BFR Q
(1A4)	420	ADDRESS	4	NCBWRKA	ADDR WORK AREA FOR DE-COM/COMPRESS
(1A8)	424	ADDRESS	1	NCBSGTE	GATE FOR SEND PER NODE
(1A9)	425	ADDRESS	1	NCBRGTE	GATE FOR RECEIVE PER NODE
		1111 1111		NCBGTIPR	"255" GATE IN PROGRESS (RETURN-CODE)
SESSION STATUS					
(1AA)	426	BITSTRING	1	NCBSFL1	PRIMARY AP
		1...		NCBF11	"X'80'" .. PRIMARY IN PROGRESS
		.1..		NCBF12	"X'40'" .. PRIMARY PERMIT GIVEN
		..1.		NCBF13	"X'20'" .. OPNDST IN PROGRESS
		...1		NCBF14	"X'10'" .. OPNDST COMPLETE
		.... 1...		NCBF15	"X'08'" .. PRIMARY COMPLETE
		1		NCBF18	"X'01'" .. PRIMARY AP ERROR
(1AB)	427	BITSTRING	1	NCBSFL2	SECONDARY AP
		1...		NCBF21	"X'80'" .. SECONDARY IN PROGRESS
		.1..		NCBF22	"X'40'" .. SECONDARY PERMIT GIVEN
		..1.		NCBF23	"X'20'" .. OPNSEC IN PROGRESS
		...1		NCBF24	"X'10'" .. OPNSEC COMPLETE
		.... 1...		NCBF25	"X'08'" .. SECONDARY COMPLETE
		.... .1..		NCBF26	"X'04'" .. SESSION IN PROGRESS
		.... ..1.		NCBSEOK	"X'02'" .. DISCONNECT REQUIRED
		1		NCBF28	"X'01'" .. SECONDARY AP ERROR
(1AC)	428	BITSTRING	1	NCBSFL3	TYPE OF SESSION EQU X'FF' .. SECONDARY HALF SESSION EQU X'00' .. PRIMARY HALF SESSION
(1AD)	429	BITSTRING	1	NCBSFL4	FLAG BYTE 4 :
		1...		NCBRSHTS	"X'80'" .. RSHUTD SENT
		.1..		NCBRSHTR	"X'40'" .. RSHUTD RECEIVED
(1AE)	430	BITSTRING	1	NCBSEST	SESSION STATUS BYTE :
		1...		NCBSSUB	"X'80'" .. UNBIND RECEIVED
		.1..		NCBSSTS	"X'40'" .. TERMSESS IS WAITING
		..1.		NCBWSDT	"X'20'" .. WAIT FOR SDT
		...1		NCBSSSD	"X'10'" .. SDT RECEIVED
		.... 1...		NCBSSCL	"X'08'" .. CLEAR RECEIVED
		.... .1..		NCBSHTC	"X'04'" .. WAIT FOR SHUTC
(1AF)	431	BITSTRING	1	NCBSSCT	SESSION RETRY COUNTER
		.... ..1.		NCBSSCTL	"X'02'" 20 MINUTES LIMIT COUNT
(1B0)	432	BITSTRING	1	NCBSSF1	SESSION FLAG BYTE
		1...		NCBSSF1W	"X'80'" .. ISSUE WAIT
		.1..		NCBSSF1R	"X'40'" .. USE RC/FD FOR MSG
(1B1)	433	BITSTRING	3		NOT USED
(1B4)	436	ADDRESS	4	NCBCMPT	COMPACTION TABLE BLOCK ADDR.
END SNA OVERLAY					
(1EC)	492	CHAR-ACTER	8	NCBCPWD	PASSWORD FOR LOCAL NODE (OUTGOING)
(1F4)	500	CHAR-ACTER	8	NCBCLPW	PASSWORD FOR LINE (OUTGOING)
		.1..		NCBLCTL	"64" LENGTH OF LINE/CTL BUFFERS

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
ACCOUNT RECORD					
(1FC)	508	SIGNED	4	NCBACCT (0)	PNET ACCOUNT RECORD
(1FC)	508	CHARACTER	8	NCBDATE	SYSTEM DATE
(204)	516	CHARACTER	4	NCBSION	SIGNON TIME (0HHMMSSF) PACKED
(208)	520	CHARACTER	4	NCBSIOF	SIGNOFF TIME (0HHMMSSF) PACKED
(20C)	524	CHARACTER	24	NCBUSER (0)	USER INFORMATION
(20C)	524	CHARACTER	8	NCBANME	NODE NAME FOR ACCOUNT RECORD
(214)	532	CHARACTER	8	NCBPWRD	NODE PASSWORD
(21C)	540	CHARACTER	8	NCBPSWD	LINE PASSWORD
(224)	548	SIGNED	2	NCBICNT	INVALID RESPONSES PER SESSION
(226)	550	CHARACTER	1	NCBRCID	PNET ACCOUNT RECORD IDENTIFIER
(227)	551	BITSTRING	1	NCBSCOD	SIGNOFF CODE
					EQU X'80' .. SIGNOFF BY OPERATOR
					EQU X'40' .. SIGNOFF BY REMOTE NODE
					EQU X'20' .. SIGNOFF DUE TO TIMEOUT
					EQU X'10' .. SIGNOFF DUE TO LINK ERROR/SESSION
					EQU X'08' .. SIGNOFF DUE TO INTERNAL ERROR
					EQU X'04' .. SIGNOFF DUE VTAM TERMINATION
					EQU X'02' .. SIGNOFF DUE VTAM HALTED BY OPER.
					EQU X'01' .. YEAR NCBSFDT IS 20YY
(228)	552	BITSTRING	1	NCBTERR	ERROR COUNT
(229)	553	CHARACTER	3	NCBDVAD	DEV ADDR/'SNA'/'TCP'/'SSL'
(22C)	556	ADDRESS	4	NCBXCNT	TRANSMISSION COUNT
(230)	560	SIGNED	2	NCBTCNT	TIMEOUT COUNT
(232)	562	SIGNED	2	NCBECNT	ERROR COUNT
(230)	560	ADDRESS	4	NCBYCNT	RECEIVE COUNT
(234)	564	CHARACTER	8	NCBSFDT	SIGNOFF DATE
		.1..		NCBACLN	"*-NCBACCT" LENGTH OF ACCOUNT RECORD
TIMER QUEUE ELEMENT					
(23C)	572	BITSTRING	24	NCBTQE (0)	TIMER QUEUE ELEMENT
(23C)	572	BITSTRING	12		NOT REFERENCED
(248)	584	BITSTRING	4	NCBEB	POST BYTES
(24C)	588	BITSTRING	8		NOT REFERENCED
EBCDIC / USASCII CODE TABLE					
(254)	596	BITSTRING	2	NCBSOHS	SOH ENQ SEQUENCE
(256)	598	BITSTRING	2	NCBSOTS	START OF TEXT SEQUENCE
(258)	600	BITSTRING	2	NCBACKS	POSITIVE ACKNOWLEDGEMENT SEQUENCE
(25A)	602	BITSTRING	2	NCBETBS (0)	DLE ETB SEQUENCE
(25A)	602	BITSTRING	1	NCBEDLE	DLE
(25B)	603	BITSTRING	1	NCBEETB	ETB
(25C)	604	BITSTRING	1	NCBNAKS	NEGATIVE ACKNOWLEDGEMENT SEQUENCE
(25D)	605	BITSTRING	3		NOT USED
FEATURE FLAGS USED FOR SIGNON CONCURRENCE					
(260)	608	BITSTRING	4	NCBIFEAT (0)	NEW FEATURE FLAGS
(260)	608	BITSTRING	1	NCBIFE1	NEW FEATURE FLAG 1
		1...		NCBIPREP	"X'80'" .. PREPARE MODE
		.1..		NCBITRM	"X'40'" .. SNA TERMINATION EXTENS.
(261)	609	BITSTRING	1	NCBIFE2	NEW FEATURE FLAG 2
(262)	610	BITSTRING	1	NCBIFE3	NEW FEATURE FLAG 3
(263)	611	BITSTRING	1	NCBIFE4	NEW FEATURE FLAG 4
(264)	612	BITSTRING	4	NCBPFEAT	WORKING FEATURE FLAGS

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(268)	616	BITSTRING	4	NCBRFEAT	RECEIVED FEATURE FLAGS
(26C)	620	BITSTRING	4	NCBSFEAT	SENT FEATURE FLAGS
(270)	624	DBL WORD	8	NCBIOT	CTC/BSC TIME OF LAST I/O
TCP/IP WORKAREA ONLY USED WITHIN NCB					
(278)	632	SIGNED	2	NCBTCPDS (0)	AREAS USED FOR TCP/IP
(278)	632	ADDRESS	4	NCBTPBF1	ADDR OF RECV BUFFER 1
(27C)	636	SIGNED	4	NCBTPBL1	LENGTH OF BUFFER 1
(280)	640	SIGNED	4	NCBTPTIS	TIMESTAMP TCP/IP PENDING
(284)	644	SIGNED	4	NCBTPTER	TIMESTAMP OF LAST ERROR
(288)	648	SIGNED	4	NCBTPTSC	TIMESTAMP FOR START CONTACT
(28C)	652	SIGNED	4		RESERVED
TCP/IP WORKAREA USED WITHIN NCB AND TDCB					
(290)	656	SIGNED	2	NCBTPDS (0)	
(290)	656	CHARACTER	15	NCBTPIPC	IP-ADDR IN READABLE FORMAT
(29F)	671	CHARACTER	1	NCBTPTYP	TYPE OF ITP WORKAREA
				NCBTPTYA	"C'A'" .. A = ACTIVE = NCB
				NCBTPTYT	"C'P'" .. P = PASSIVE = TDCB
				NCBTPTYC	"C'C'" .. C = CLIENT, WA IN NCB
				NCBTPTYS	"C'X'" .. S = SERVER, WA IN NCB
START OF AREA-1 TO BE TRACED					
(2A0)	672	BITSTRING	36	NCBTPTC1 (0)	START OF TRACED INFO 1
(2A0)	672	BITSTRING	1	NCBTPST1	TCP/IP STATUS BYTE 1:
GENERAL TCP/IP STATUS, SOCKETCALL STATUS					
		1...		NCBTPS1T	"X'80'" .. TCP/IP INIT CONTACT COMPL
		.1..		NCBTPS1F	"X'40'" .. TCP/IP CONN. CLOSED
		..1.		NCBTPS1R	"X'20'" .. TCP/IP RESTART: NAK-3
		...1		NCBTPS1E	"X'10'" .. TCP/IP LINE ERROR
		.... 1...		NCBTPS1A	"X'08'" .. PROCESSING ACTIVE MODE
		.... .1..		NCBTPS1I	"X'04'" .. 1.SOCKETCALL ISSUED
		.... ..1.		NCBTPS1L	"X'02'" .. SSL FEATURE INITIATED
		1		NCBTPS1S	"X'01'" .. STOP CONNECTION
(2A1)	673	BITSTRING	1	NCBTPST2	TCP/IP STATUS BYTE 2
GENERAL NODE STATUS					
		1...		NCBTPS2I	"X'80'" .. CTC I/O ONCE PROCESSED
		.1..		NCBTPS2R	"X'40'" .. RESTART TCP/IP PV010222
		..1.		NCBTPS2B	"X'20'" .. FIRST COMES TTB
		...1		NCBTPS2C	"X'10'" .. CLOSE CONNECTION
		.... 1...		NCBTPS2O	"X'08'" .. OPEN-CTRL-REC. RECEIVED
		.... .1..		NCBTPS2A	"X'04'" .. ACK-CTRL-REC. SENT
		.... ..1.		NCBTPS22	"X'02'" .. NAK WITH RC=2 SENT
		1		NCBTPS2W	"X'01'" .. WAIT THAT REMOTE ISSUES .. .. CONNECT
(2A2)	674	BITSTRING	1	NCBTPST3	TCP/IP STATUS BYTE 3
STATUS: CTC I/O					
		1...		NCBTPS3S	"X'80'" .. CTC I/O STARTED
		.1..		NCBTPS3C	"X'40'" .. CTC I/O TO BE COMPLETED
		..1.		NCBTPS3Z	"X'20'" .. CCW-WRITE DATA SENT
		...1		NCBTPS3Y	"X'10'" .. CCW-READ DATA RECVED
		.... 1...		NCBTPS3B	"X'08'" .. CTC I/O WITHOUT BUFFER
		.... .1..		NCBTPS3N	"X'04'" .. TCP BLOCK PARTLY RCVED
		.... ..1.		NCBTPS3L	"X'02'" .. LEAVE IDLING STATE
		1		NCBTPS3I	"X'01'" .. IDLING(NOTHING SENT/RCV)
(2A3)	675	BITSTRING	1	NCBTPST4	TCP/IP STATUS BYTE 4
STATUS: MISCELLANEOUS					
		1...		NCBTPS4P	"X'80'" .. WAIT FOR POSTED ECB
			.1..		NCBTPS4C	"X'40'" .. CONNECTION CLOSED

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
		..1.		NCBTPS4T	"X'20'" .. TERMINATE LINE
		...1		NCBTPS4W	"X'10'" .. WAIT TILL TIME EXPIRED
		.... 1...		NCBTPS4A	"X'08'" .. CANCEL ISSUED
		.... .1..		NCBTPS4L	"X'04'" .. CLOSE ISSUED
		.... ..1.		NCBTPS4N	"X'02'" .. SOK NUMBER TOO HIGH (SSL)
		1		NCBTPS4E	"X'01'" .. SEND EMPTY BUFFER
(2A4)	676	BITSTRING	1	NCBTPST5	TCP/IP STATUS BYTE 5:
CLOSING CODES, CLOSED DUE TO:					
		1...		NCBTPS5U	"X'80'" .. SIGNOFF REC. SEND
		.1..		NCBTPS5R	"X'40'" .. SIGNOFF REC. RECEIVED
		..1.		NCBTPS5A	"X'20'" .. INVALID DEFINITION
		...1		NCBTPS5N	"X'10'" .. TCP NJE NAK RECEIVED
		.... 1...		NCBTPS58	"X'08'" .. CAUSED BY TCP/IP RC=8
		.... .1..		NCBTPS5C	"X'04'" .. CAUSED BY TCP/IP RC=12
		.... ..1.		NCBTPS5I	"X'02'" .. POWER INTERNAL ERROR
		1		NCBTPS5S	"X'01'" .. CAUSED BY REMOTE CLOSED
(2A5)	677	BITSTRING	1	NCBTPST6	TCP/IP STATUS BYTE 6:
		1...		NCBTPS6T	"X'80'" .. TRACE SOCKETCALL
		.1..		NCBTPS6I	"X'40'" .. INIT TIME INTERVAL
		..1.		NCBTPS6C	"X'20'" .. SSL-SOK-INIT: CIPH WRONG
(2A6)	678	BITSTRING	1	NCBTPRV1	RESERVED
(2A7)	679	BITSTRING	1	NCBTPRV2	RESERVED
RETURN ADDRESSES FOR SOCKETCALL ROUTINE					
(2A8)	680	ADDRESS	4	NCBTPSO0	SOCKETCALL SUCCESSFUL
(2AC)	684	ADDRESS	4	NCBTPSO4	SOCKETCALL SHOULD BE RETRIED
(2B0)	688	ADDRESS	4	NCBTPSO8	CONNECTION TO BE STOPPED
(2B4)	692	ADDRESS	4	NCBTPSOC	INTERFACE TO BE TERMINATED
INTERFACE AREA BETWEEN IPW$$TD AND IPW$$TS RESPECTIVELY BETWEEN IPW$$SD AND IPW$$SS					
(2B8)	696	SIGNED	4	NCBTPR1	RETURN CODE FROM $TS
		....		NCBTPR10	"0" .. OK
		.... .1..		NCBTPR14	"4" .. RETRY NECESSARY/POSSIBLE
		.... 1...		NCBTPR18	"8" .. TERMINATE CONNECTION
		.... 11..		NCBTPR1C	"12" .. TERMINATE INTERFACE
		.... ..1.		NCBTPR1R	"2" .. RETRY DUE TO IPW$$SD
(2BC)	700	BITSTRING	1	NCBTPSC	SOCKETCALL REQUESTED
		1		NCBTPIA	"1" .. INITAPI
		.... ..1.		NCBTPTA	"2" .. TERMAPI
		11		NCBTPGL	"3" .. GETHOSTID
		.... .1..		NCBTPLI	"4" .. LISTEN
		.... .1.1		NCBTPAC	"5" .. ACCEPT
		.... .11.		NCBTPSD	"6" .. SEND
		.... .111		NCBTPRV	"7" .. RECEIVE
		.... 1...		NCBTPCL	"8" .. CLOSE
		.... 1..1		NCBTPCN	"9" .. CANCEL
		.... 1.1.		NCBTPGA	"10" .. GETHOSTBYADDR
		.... 1.11		NCBTPGN	"11" .. GETHOSTBYNAME
		.... 11..		NCBTPGS	"12" .. GET SOCKET
		.... 11.1		NCBTPBI	"13" .. BIND
		.... 111.		NCBTPCO	"14" .. CONNECT
		 1111		NCBTPSR	"15" .. SELECT USING READ-ARRAY
		...1		NCBTPSW	"16" .. SELECT USING WRITE-ARRAY
SOME SSL SOCKET CALLS:					
		...1 ...1		NCBSSLIN	"17" .. SSL INITIALIZE
		...1 ..1.		NCBSSLUN	"18" .. SSL UNINITIALIZE
		...1 ..11		NCBSSLGN	"19" .. SSL GET DNAME IN DB
		...1 .1..		NCBSSLFM	"20" .. SSL FREE MEMORY
		...1 .1.1		NCBSSLSI	"21" .. SSL SOCKET INITIALIZE
		...1 .11.		NCBSSLSR	"22" .. SSL SOCKET READ
		...1 .111		NCBSSLSW	"23" .. SSL SOCKET WRITE
		...1 1...		NCBSSLSC	"24" .. SSL SOCKET CLOSE

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
		...1 1..1		NCBSSLRS	"25" .. SSL SOCKET RESET
		...1 1.1.		NCBSSLGC	"26" .. SSL GET CIPHER INFO
		...1 1.11		NCBTPIOC	"27" .. IOCTL=SET NONBLOCKING
(2BD)	701	BITSTRING	1		UNUSED
(2BE)	702	BITSTRING	2	NCBTPRYC	RETRY COUNTER
(2C0)	704	SIGNED	4	NCBTPTIV	TIMER INTERVAL TO BE SET
		..1. .1..		NCBTPTE1	"*-NCBTPTC1" LENGTH OF TRACED INFO 1
(2C4)	708	BITSTRING	24	NCBTPTQE (0)	TIMER QUEUE ELEMENT
(2C4)	708	BITSTRING	12		NOT REFERENCED
(2D0)	720	BITSTRING	4	NCBTPTEB	POST BYTES
(2D4)	724	BITSTRING	5		NOT REFERENCED
(2D9)	729	CHARACTER	3	NCBTPTQY	EYECATCHER
AREAS USED FOR SOCKET CALLS AREAS USED FOR SEVERAL SOCKET CALLS START OF AREA-2 TO BE TRACED					
(2DC)	732	ADDRESS	4	(0)	ALIGN
(2DC)	732	BITSTRING	20	NCBTPTC2 (0)	START OF TRACED INFO 2
(2DC)	732	SIGNED	2	NCBSCSOD	SOCKET DESCRIPTOR
(2DE)	734	SIGNED	2		RESERVED
AREAS USED FOR SOCKET CALLS: BIND, ACCEPT, CONNECT					
(2E0)	736	BITSTRING	16	NCBSCDNM (0)	
(2E0)	736	SIGNED	2	NCBSCBFM	ADDRESSING FAMILY
(2E2)	738	SIGNED	2	NCBSCBPT	PORT NUMBER
(2E4)	740	SIGNED	4	NCBSCDIP	IP-ADDRESS
(2E8)	744	BITSTRING	4		RESERVED FOR SOCKETCALL
(2EC)	748	BITSTRING	4		RESERVED FOR SOCKETCALL
		...1 .1..		NCBTPTE2	"*-NCBTPTC2" LENGTH OF TRACED INFO 2
AREAS USED FOR SEVERAL SOCKET CALLS EXCEPT SEND AND CANCEL					
(2F0)	752	ADDRESS	4	(0)	ALIGN
(2F0)	752	BITSTRING	1	NCBSCST1	STATUS OF SOCKETCALL
		1...		NCBSCS1S	"X'80'" .. SOCKETCALL STARTED
		.1..		NCBSCS1B	"X'40'" .. NO BUFFER AVAILABLE
(2F1)	753	BITSTRING	1		RESERVED
(2F2)	754	SIGNED	2	NCBSCCNT	RETRY COUNTER
(2F4)	756	ADDRESS	4	NCBSCBUF	BUFFER FOR RECV
(2F8)	760	ADDRESS	4	NCBSCNBY	NO OF BYTES FOR RECV
(2FC)	764	SIGNED	4	NCBSCRC	RETURN CODE FROM SOCKETCALL
(300)	768	SIGNED	4	NCBSCERN	ERROR NUMBER
(304)	772	BITSTRING	164	NCBSCDCB (0)	
(304)	772	ADDRESS	4	NCBSCECB (0)	ECB
(304)	772	BITSTRING	2		.. UNREFERENCED
(306)	774	BITSTRING	1	NCBSCECP	.. POSTED BYTE
		1...		NCBSCECI	"X'80'" .. POST BIT
(307)	775	BITSTRING	1		.. UNREFERENCED
(308)	776	BITSTRING	160	NCBSCRQ	WORKAREA FOR EZASMI
AREAS USED FOR SOCKET CALL : INITAPI, LISTEN SEE TDCB AREAS USED FOR SOCKET CALLS: GETHOSTBYADDR					
(3A8)	936	ADDRESS	4	NCBSCHST	ADDR. OF HOSTNAME STRUCTURE
AREAS USED FOR SOCKET CALL : GETHOSTBYNAME AND AREA ..SCHST, SEE SOCKET CALL GETHOSTBYADDR					
(3AC)	940	SIGNED	4	NCBSCHNL	LENGTH OF HOSTNAME
(3B0)	944	ADDRESS	4	NCBSCHNM	ADDRESS OF HOSTNAME

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
AREAS USED FOR SOCKET CALL: SEND					
(3B4)	948	ADDRESS	4	(0)	ALIGN
(3B4)	948	BITSTRING	8	NCBTPTC3 (0)	START OF TRACED INFO 3
(3B4)	948	BITSTRING	1	NCBSCSS1	STATUS OF SOCKETCALL
		1...		NCBSCD1S	"X'80'" .. SOCKETCALL STARTED
		.1..		NCBSCD1B	"X'40'" .. NO BUFFER AVAILABLE
(3B5)	949	BITSTRING	1		RESERVED
(3B6)	950	SIGNED	2	NCBSCSCT	RETRY COUNTER
(3B8)	952	ADDRESS	4	NCBSCBUS	SOCKETCALL TO BE CANCELLED
(3BC)	956	ADDRESS	4	NCBSCNBS	UNREFERENCED
(3C0)	960	SIGNED	4	NCBSCSAC	RETURN CODE
(3C4)	964	SIGNED	4	NCBSCSAE	ERROR NUMBER
		...1 .1..		NCBTPTE3	"*-NCBTPTC3" LENGTH OF TRACED INFO 3
(3C8)	968	BITSTRING	164	NCBSCSAL (0)	
(3C8)	968	ADDRESS	4	NCBSCSAB (0)	ECB
(3C8)	968	BITSTRING	2		.. UNREFERENCED
(3CA)	970	BITSTRING	1	NCBSCSAP	.. POSTED BYTE
(3CB)	971	BITSTRING	1		.. UNREFERENCED
(3CC)	972	BITSTRING	160	NCBSCSAR	WORKAREA FOR EZASMI
AREAS USED FOR SOCKET CALL: CANCEL PARTLY RE-USED FOR SOCKET CALL: GET CIPHER INFO					
(46C)	1132	ADDRESS	4	(0)	ALIGN
(46C)	1132	BITSTRING	1	NCBSCCS1	STATUS OF SOCKETCALL
		1...		NCBSCC1S	"X'80'" .. SOK CALL STARTED, UNUSED
		.1..		NCBSCC1B	"X'40'" .. NO BFR AVAILABLE, UNUSED
(46D)	1133	BITSTRING	1		RESERVED
(46E)	1134	SIGNED	2	NCBSCCCT	RETRY COUNTER
(470)	1136	ADDRESS	4	NCBSCBUC	SOCKETCALL TO BE CANCELLED
(474)	1140	ADDRESS	4	NCBSCNBC	UNREFERENCED
(478)	1144	SIGNED	4	NCBSCCAC	RETURN CODE
(47C)	1148	SIGNED	4	NCBSCCAE	ERROR NUMBER
(480)	1152	BITSTRING	164	NCBSCCAL (0)	
(480)	1152	ADDRESS	4	NCBSCCAB (0)	ECB
(480)	1152	BITSTRING	2		.. UNREFERENCED
(482)	1154	BITSTRING	1	NCBSCCAP	.. POSTED BYTE
(483)	1155	BITSTRING	1		.. UNREFERENCED
(484)	1156	BITSTRING	160	NCBSCCAR	WORKAREA FOR EZASMI
PARTLY RE-USED FOR SOCKET CALL: GET-CIPHER-INFO					
(480)	1152	BITSTRING	104	NCBSSLCO (0)	OUTPUT OF GET-CIPHER-INFO
(480)	1152	SIGNED	4		SYSTEM SSL VERSION
(484)	1156	BITSTRING	64	NCBSSLCC	SPECS OF GET-CIPHER-INFO
(4C4)	1220	BITSTRING	30		INPUT FOR SSL-SOK-INIT
(4E2)	1250	CHARACTER	6		ENCRYPTION IN CHARACTERS
(524)	1316	SIGNED	4		RESERVED
NO SPECIAL AREAS FOR SOCKET CALLS: CLOSE, SOCKET, GETHOSTID, TERMAPI SEE AREAS USED FOR SEVERAL SOCKET CALLS AREAS USED FOR SSL SOCKET CALLS					
(528)	1320	ADDRESS	4	NCBSSLCB	ADDR OF SSL CONTROL BLOCK RETURNED BY SSL-SOCK-INIT
(52C)	1324	SIGNED	4	NCBSSLRC	REASON-CODE OF SOCK-INIT
(530)	1328	ADDRESS	4	NCBSSLDN	ADDR OF DISTINGUISHED NAME RETURNED BY SSL-GETDNBYLAB INPUT FOR SSL-FREEMEM
(534)	1332	ADDRESS	4	NCBSSLCF	ADDR OF CLIENT-CERTIFICATE UPDATED BY SSL-SOCK-INIT

Offset	Offset	Type	Len	Name (Dim)	Description
Hex	Dec				
(538)	1336	ADDRESS	4	NCBSSLST	ADDR OF CLIENT-CERTIFICATE UPDATED BY SSL-SOCK-INIT
(53C)	1340	SIGNED	2	NCBSSLCS	2 BYTES OF SELECTED CIPHERS IS PART WITHIN ---SLCPO
(53E)	1342	SIGNED	2		RESERVED
(540)	1344	SIGNED	4	NCBIOCMD	MODE FOR IOCTL
		....		NCBIOCBL	"0" .. BLOCKING MODE
		1		NCBIOCNB	"1" .. NONBLOCKING MODE
(544)	1348	SIGNED	4		RESERVED
(548)	1352	SIGNED	4		RESERVED
(54C)	1356	SIGNED	4		RESERVED
(550)	1360	SIGNED	4		RESERVED
(554)	1364	SIGNED	4		RESERVED
(558)	1368	SIGNED	4		RESERVED
(55C)	1372	SIGNED	4		RESERVED
(560)	1376	SIGNED	4		RESERVED
(564)	1380	SIGNED	4		RESERVED
(568)	1384	SIGNED	4		RESERVED
(56C)	1388	SIGNED	4		RESERVED
WORKAREA FOR IPW$$TD, RESPECTIVELY IPW$$SD					
(570)	1392	SIGNED	4	NCBTPNOB	NO OF BYTES SEND/RCVED
(574)	1396	BITSTRING	33	NCBTPCTB	BUFFER FOR CTRL-RECORD
(595)	1429	BITSTRING	3	NCBTPCR1	RESERVED
(598)	1432	SIGNED	4	NCBTPBR1	BYTES RCVED VIA SOCKETCALL
(59C)	1436	SIGNED	4	NCBTPBP1	BYTES PROCESSED BY .. .. IPW$$TD, RESP. IPW$$SD
(5A0)	1440	ADDRESS	4	NCBTPWPO	ADDRESS: WAIT FOR POST ECB
(5A4)	1444	ADDRESS	4	NCBTPNCB	ADDRESS OF NCB
(5A8)	1448	BITSTRING	2		RESERVED
(5AA)	1450	BITSTRING	2	NCBTPFCS	FCS SAVED FROM CTC I/O
(5AC)	1452	BITSTRING	1		RESERVED
(5AD)	1453	BITSTRING	1	NCBTPBCS	BCB SAVED FROM CTC I/O
(5AE)	1454	BITSTRING	1	NCBTPBCI	BCB FOR INCOMING BUFFER
(5AF)	1455	BITSTRING	1	NCBTPBCO	BCB FOR OUTGOING BUFFER
(5B0)	1456	SIGNED	4		RESERVED
(5B4)	1460	SIGNED	4		RESERVED
(5B8)	1464	SIGNED	4		RESERVED
(5BC)	1468	SIGNED	4		RESERVED
(5C0)	1472	SIGNED	4		RESERVED
(5C4)	1476	SIGNED	4		RESERVED
(5C8)	1480	SIGNED	4		RESERVED
(5CC)	1484	SIGNED	4		RESERVED
(5CC)	1484		0	NCBTPLST	"*" END OF WORKAREA
(5CC)	1484		0	NCBTPLN	"*-NCBTPDS" LENGTH OF WORKAREA
AREA NOT TO BE CLEARED AFTER SOCKET CALL CLOSE					
(5D0)	1488	SIGNED	4	NCBSSLDS (0)	' START OF SSL WORKAREA
(5D0)	1488	CHARACTER	8	NCBSSLKY	MEMBER IN SUBLIB
(5D8)	1496	BITSTRING	1		END DELIMITER FOR DNAME
(5D9)	1497	BITSTRING	1		RESERVED FOR FUTURE USE
(5DA)	1498	SIGNED	2		RESERVED FOR FUTURE USE
(5DC)	1500	SIGNED	4	NCBSSLHK	HANDSHAKE TYPE
		....		NCBSSLHC	"0" HANDSHAKE TYPE: CLIENT
		1		NCBSSLHS	"1" HANDSHAKE TYPE: SERVER
		.... ..1.		NCBSSLHA	"2" HANDSHAKE TYPE: CLIENT AUTH
		11		NCBSSLHN	"3" HANDSHAKE TYPE: NO CLI AUTH
(5E0)	1504	SIGNED	4	NCBSSLCP	CIPHER LEVEL
		1		NCBSSLCL	"1" CIPHER LEVEL: WEAK
		.... ..1.		NCBSSLCH	"2" CIPHER LEVEL: STRONG
		11		NCBSSLEV	"3" CIPHER LEVEL: NORMAL
(5E4)	1508	ADDRESS	4		RESERVED FOR FUTURE USE
(5E8)	1512	ADDRESS	4		RESERVED FOR FUTURE USE
(5EC)	1516	ADDRESS	4		RESERVED FOR FUTURE USE

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(5F0)	1520	ADDRESS	4		RESERVED FOR FUTURE USE
(5F4)	1524	ADDRESS	4		RESERVED FOR FUTURE USE
(5F8)	1528	ADDRESS	4		RESERVED FOR FUTURE USE
(5FC)	1532	ADDRESS	4		RESERVED FOR FUTURE USE
		..11		NCBSSLDL	"*-NCBSSLDS" LENGTH OF WORKAREA
(5FC)	1532		0	NCBLN	"*-NCBDS" LENGTH OF CONTROL BLOCK

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
NODE CONTROL BLOCK TASK ENTRY					
(0)	0	STRUCTURE	0	NCBEDS	, DUMMY SECTION DEFINITION
(0)	0	BITSTRING	1	NCBERCB	RCB OF TASK CONCERNED
(1)	1	BITSTRING	1	NCBEST1	STATUS/ACTION BYTE
		1...		NCBTDRN	"X'80'" .. TASK DRAINED
		.1..		NCBTLVE	"X'40'" .. TASK LIVE
		..1.		NCBDETE	"X'20'" .. DEQUEUE & DELETE NCB TASK ENTRY
		...1		NCBTCRE	"X'10'" .. TASK CREATION REQUESTED
(2)	2	BITSTRING	1	NCBESTS	TASK STOP STATE (DUPLICATE TO TCB)
(3)	3	BITSTRING	1	NCBETYP	TASK TYPE
		1...		NCBETYPT	"X'80'" .. TRANSMITTER TASK
		.1..		NCBETYPR	"X'40'" .. RECEIVER TASK
		..1.		NCBETYPC	"X'20'" .. CONSOLE TASK
		.... 1...		NCBETYPJ	"X'08'" .. JOB PROCESSING
		.... .1..		NCBETYPO	"X'04'" .. OUTPUT (LST,PUN) PROCESSING
(4)	4	ADDRESS	4	NCBETCB	TCB ADDRESS OF TASK
		.... 1...		NCBELN	"*-NCBEDS" LENGTH OF ONE ENTRY

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
PASSED PARAMETERS TO IPW$$LD5 DUE TO RESTRUCTERING OF PHASES					
(0)	0	STRUCTURE	0	PASPARD	, DUMMY SECTION DEFINITION
(0)	0	BITSTRING	12	PASPAR5	PARAMETERS FOR IPW$$LD5
(C)	12	BITSTRING	1	PASTIME	TIME PARAMETER
(D)	13	BITSTRING	2		NOT USED
(F)	15	BITSTRING	1	PASFCT5	FUNCTION FOR IPW$$LD5
					EQU X'01' TEST RCB OF RIF
					EQU X'02' TEST RCB FOR TRANSMITTER
					EQU X'03' TEST RCB FOR RECEIVER
					EQU X'04' TEST LOG ERROR ON SYSREC
					EQU X'05' TEST CANCEL TASKS
					EQU X'06' TEST CREATE TASK
					EQU X'07' TEST UPDATE NAT
					EQU X'08' TEST SET TIMER-INTERVAL
		...1		PASPLN	"*-PASPARD" LENGTH OF ONE ENTRY

Node Control Block Task Entry

Definition Macro: IPW$DNC

(see NCB)

Open 3540 Diskette Work Space

DSECTname: OEWS (Module IPW$$OE)

This work area is used during the opening of 3540 diskette files.

Bytes Hex.	Label of Field	Description/Function
00-0F		Storage descriptor 'OEWS REL cuu'
	OECB	3540 command control block
10-11	OECT	Residual count
12-13	OECM	Communications bytes
14-15	OEST	Device status
16-17	OELU	Device type and logical unit
18		Reserved for LIOCS
19-1B	OECA	First CCW
1C		Reserved for PIOCS
1D-1F	OECW	CCW address in CSW
20-57	OESV	Temporary register save area for the interface between functions
58-5F	OECV	Conversion work space
	OECP	3540 channel program
60-67	OEDO	Define operations or NOP
68-6F	OESK	Seek
70-77	OERD	Read label
78-7B	OESM	Mode setting argument
7C-7F	OESA	Seek argument (00CCHHRR)
80-CF	OELB	3540 input area and label test area (see below)

- Message Buffers and Work Areas

D0	OML1	Message length of first line
	OMS1	First line of message output area
D1-D7	OMT1	Message identity
D8-107	OMT1	Message text of first line
108	OML2	Message length of second line
	OMS2	Second line of message output area
109	OMI2	Message identity
110-13F	OMT2	Message text of second line
140		Not used
141	OERL	Reply length
142-147	OERP	Reply input area
148	OECC	Cylinder number save area

- Physical reader information indicators. The following indicators are copied from the physical work space to prevent them from being destroyed should the OPEN be unsuccessful. On a successful OPEN, the indicators in the physical work space are overwritten by these updated indicators. On an unsuccessful OPEN, only the OPEN indicator 'PEOC' will be updated with the stop code 'S'. (See "Physical Work Space" in this chapter.)

Bytes Hex.	Label of Field	Description/Function
14A-14B	WERL	Record length (copy of PERL)
	WESI	Sequence ID (copy of PESI)
14C	WEMI	Multi-volume identification (copy of PEMI)
14D	WESN	Volume sequence number (copy of PESN)
14E	WEOD	Number of opened diskettes (copy of PEOD)
14F	WEND	Number of diskettes to be read (copy of PEND)
150-157		Not used

- 3540 Volume 1 Label Layout in Label Test Area (OELB)

	VOLL	Diskette volume 1 label
80-83	VLID	Volume label ID and number
84-89	VLSN	Volume serial number
8A	VLAI	Volume access indicator
8B-A4		Reserved
A5-B2	VLDI	Volume owner identity
B3-CA		Reserved
CB	VLPL	Physical record length
CC-CD	VLRS	Physical record sequence code
CE		Reserved
CF	VLST	Label standard version (W)

- 3540 Header 1 Label Layout in Label Test Area (OELB)

	HDRL	Diskette header 1 label
80-83	HDID	Header label ID and number
84		Reserved
85-8C	HDFI	File identifier
8D-95		Reserved
96-9A	HDBL	Block length of data record
9B		Reserved
9C-A0	HDLO	Begin of extent (CCHRR)
A1		Reserved
A2-A6	HDHI	End of extent (CCHRR)
A7		Reserved
A8	HDBI	Bypass indicator (B)
A9	HDFS	File security indicator (S)
AA	HDWP	File write protection indicator (P)
AB	HDEI	Basic exchange indicator (,E)
AC	HDMV	Multi-volume indicator (,C,L)
AD-AE	HDSN	Volume sequence number
AF-B4	HDCR	Creation date
B5-C1		Reserved
C2-C7	HDEX	Expiration date
C8	HDVI	Verify indicator (,V)
C9		Reserved
CA-CE	HDED	End of data address (CCHRR)
CF		Reserved

Output Exit Parameter List

Definition Macro: IPW$DXE

This macro is used to produce a DSECT for the Output Exit Parameter List. The format is as follows:

Bytes Hex.	Label of Field	Description/Function
00	OEXDS	Start of DSECT
00-03	OEXRV	Record address of statement passed
04-07	OEXRL	Length of statement passed
08	OEXCC	Operation code
09	OEXRT	Record type
	OERNCD	X'00' - Normal data or control record
	OERJHR	X'80' - Job header record
	OERJTR	X'40' - Job trailer record
	OERDSHR	X'20' - Data set header record
	OERSEP	X'08' - Record of start separator page
	OERESEP	X'04' - Record of end separator page
0A	OEXTT	Task type
	OETLST	X'80' - List task
	OETPUN	X'40' - Punch task
	OETRJE	X'20' - RJE task
	OETDST	X'02' - Device service task
0B	OEXOT	Various information
	OEOLST	X'80' - Output from list queue
	OEOPUN	X'40' - Output from punch queue
	OEOSQE	X'20' - Start of queue entry
	OEOSNC	X'10' - Start next copy
	OEOQEP	X'08' - Queue entry processed
	OEOSPA	X'04' - PSETUP command active
0C-0F	OEXWA	Pointer to exit work area
10	OEXRC	Return codes
	OEROK	X'00' - Normal processing
	OERDEL	X'04' - Delete this record
	OERINS	X'08' - Insert new record
	OERFLS	X'10' - Flush queue entry
	OERFLH	X'18' - Flush hold queue entry
	OERSTP	X'1C' - Stop task
11-13		Reserved for future use

Output Parameter Definition Entry

Definition Macro: IPW$DOP OPDE=YES

This macro is used to produce a DSECT for the Output Parameter Definition Entry. The format is as follows:

Bytes Hex.	Label of Field	Description/Function
00	OPDEDS	Start of DSECT
00-03	OPNXT	Pointer to next OPDE in chain
04-05	OPDI	Registered keyword identifier
06	OPCAR	Carrier type
	OPCARLST	C'L' - Carrier is LST statement
	OPCARPUN	C'P' - Carrier is PUN statement
07	OPKWLN	Length of keyword
08-0F	OPKW	Keyword (output parameter)
10-11	OPREPEAT	Maximun repeat (=number of subparameters)
12-13	OPVALLNT	Maximum length of keyword value
14-16		Reserved for future use
17	OPVALTYP	Type of keyword value
	OPVTANY	C'*' - Any character allowed
	OPVTALPH	C'A' - Alphabetic value
	OPVTCHAR	C'C' - Alphameric value
	OPVTNUM	C'N' - Numeric value
	OPVTHEX	C'H' - Hexadecimal value
	OPVTBIN	C'B' - Binary value
18-1B	OPMINVAL	Minimum value for a binary value
1C-1F	OPMAXVAL	Maximum value for a binary value

Output Parameter Text Block

Definition Macro: IPW$DOP OPTB=YES

This macro is used to produce a DSECT for the Output Parameter Text Block. The format is as follows:

Bytes Hex.	Label of Field	Description/Function
00	OPTBHDS	DSECT for OPTB header
00-01	OPTBID	Registered keyword identifier
02-03	OPTBCNT	Number of data elements
00	OPTBDDS	DSECT for OPTB data element
00-01	OPTBDLEN	Length of data element
02	OPTBDVAL	Start of data element value

Output Parameter Processing Interface List

Definition Macro: IPW$DOP OPI=YES

This macro is used to produce a DSECT for the Output Parameter Processing Interface List. The format is as follows:

Bytes Hex.	Label of Field	Description/Function
• OPI General Section		
00	OPIDS	Start of DSECT
00-17F	OPIDWA	Dynamic work area for IPW$$OP
180	OPIFUNC	Function key
	OPIFBLD	X'01' - Build OPDE chain
	OPIFANAL	X'02' - Analyze output parameter
	OPIFPUT	X'03' - Specify OPTBs
	OPIFGET	X'04' - Retrieve OPTBs
	OPIFMOD	X'05' - Modify one OPTB
181	OPIRC	Return codes set by IPW$$OP
	OPIRCOK	X'00' - Ok, no error occurred
	OPIRCIDF	X'01' - Invalid parameter in DEFINE statement
	OPIRCJER	X'02' - JECL error
	OPIRCIOP	X'03' - Invalid OPTB
	OPIRCDOP	X'04' - Duplicate OPTB
	OPIRCONF	X'05' - OPTB not found
	OPIRCRTS	X'06' - Return area too small
	OPIRCOLM	X'07' - OPTB length mismatch
	OPIRCOTL	X'08' - Specified OPTBs too long
	OPIRCIDH	X'09' - Invalid Data Set Header Record
	OPIRCNST	X'10' - No storage available
	OPIRCINK	X'11' - Keyword syntax invalid
	OPIRCNDK	X'12' - No DEFINE statement found for keyword
	OPIRCIDV	X'13' - Keyword value invalid
182-183		Reserved for future use
184	OPIGEND	End of general section
• OPDEBLD Interface Section		
184-187	OPBDEFIN	Address if area with DEFINE statement
188	OPBMSGRC	Reason code for message 1Q09I
	OPBMOK	X'00' - No reason code required
	OPBMCAR	X'01' - Invalid carrier type
	OPBMKW	X'02' - Invalid keyword
	OPBMDUKW	X'03' - Duplicate keyword
	OPBMID	X'04' - Invalid identifier
	OPBMDUID	X'05' - Duplicate identifier
	OPBMREP	X'06' - Invalid repeat factor
	OPBMLN	X'07' - Invalid length specification
	OPBMTYPE	X'08' - Invalid type specification
	OPBMNAPP	X'09' - Minimum/Maximum value not applicable
	OPBMMIN	X'10' - Invalid minimum value specified
	OPBMMAX	X'11' - Invalid maximum specified
	OPBMMIMA	X'12' - Minimum greater than maximum
	OPBMTOOM	X'13' - Too many parameters specified
	OPBMDLIM	X'14' - Invalid statement delimiter
	OPBMCONT	X'15' - Continuation not allowed

Bytes Hex.	Label of Field	Description/Function
• OPANAL Interface Section		
184-187	OPADSHR	Address of Data Set Header record
188-18B	OPAPARA	Address of parameter string to be analyzed
18C-18F	OPASTRT	Start address of JECL statement
190-193	OPAEND	End address of JECL statement
194-197	OPAOWNER	TCB address of task owning DSHR
198	OPACAR	Carrier type
199		Reserved for future use
19A-19B	OPAMSG	Message indicator ($1Q50I,$1Q51I)
19C-19F	OPAMSGKW	Address of keyword to be included into message 1Q51I
1A0-1A3	OPADELIM	Pointer to delimiter of parameter string
• OPPUT Interface Section		
184-187	OPPDSHR	Address of Data Set Header
188-18B	OPPCDPT	Address of code point area containing OPTBs to be added
18C-18D	OPPCDLEN	Length of code point area
18E	OPPCAR	Carrier type
	OPPCPUN	C'P' - Punch queue entry
	OPPCLST	C'L' - List queue entry
• OPGET Interface Section		
184-187	OPGDSHR	Address of Data Set Header, if storage copy exists
188-18B	OPGRETAD	Address of area where OPTBs should be returned
18C-18D	OPGRETLN	Length of return area
18E-18F	OPGKWID	Keyword identifier of OPTB
190-191	OPGACTLN	Actual length of returned OPTB(s)
• OPMOD Interface Section		
184-187	OPMDSHR	Address of Data Set Header, if storage copy exists
188-18B	OPMNEWAD	Address of new version of OPTB to be replaced
18C-18D	OPMNEWLN	Length of new OPTB
18E	OPMCAR	Carrier type
	OPMCPUN	C'P' - Punch queue entry
	OPMCLST	C'L' - List queue entry

Partition Control Block (PDB)

Definition Macro: IPW$DPD

A partition control block is created for each partition to be controlled by VSE/POWER. In addition to general partition information, the block contains an entry for each device that is to be spooled. The format of these entries is described by the IPW$DDE macro. The partition control blocks for static partitions are located just behind the Nucleus in the SVA part of VSE/POWER and are allocated during initialization of VSE/POWER. Each contains place for 29 (maximum) spool devices.

The partition control blocks for the dynamic partitions are also located in the SVA and allocated during partition allocation. Each contain placeholders for the number of spool devices specified in the dynamic class table.

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
					PARTITION CONTROL BLOCK (PDB)

THE FIRST PART OF THE BLOCK CONTAINS GENERAL PARTITION INFORMATION, STATISTICAL INFORMATION DESTINED FOR THE EXECUTION ACCOUNT RECORD AND OTHER CONTROL INFORMATION.

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(0)	0	CHARACTER	16	PSSD	STORAGE DESCRIPTOR
(10)	16	CHARACTER	2		RESERVED
(12)	18	CHARACTER	2	PDPI	PARTITION SYSLOG ID
(14)	20	SIGNED	4	PDNE	NUMBER OF ENTRIES
(18)	24	ADDRESS	4	PDCM	PARTITION COMREG
(1C)	28	ADDRESS	4	PDPCE	PCE ADDRESS
(20)	32	ADDRESS	4	PDPA	FIRST ENTRY ADDRESS
(24)	36	ADDRESS	4	PDBA	PARTITION VIRT.BEGIN ADDR.
(28)	40	ADDRESS	4	PDEA	PARTITION VIRT.END ADDR.
(2C)	44	ADDRESS	4	PDRL	PARTITION REAL BEGIN ADDR.
(30)	48	ADDRESS	4	PDRH	PARTITION REAL END ADDR.
(34)	52	BITSTRING	2	PDJN	LST/PUN JOB NR INDICATORS
(36)	54	BITSTRING	1	PDTT	TERMINATION CODE
(37)	55	BITSTRING	1	PDFLG	FLAG BYTE 1
		1...		PDFHLDI	"X'80'" 'PAUSE' REQUEST FROM INIT.
		.1..		PDFHLDX	"X'40'" 'PAUSE' REQUEST FOR DISP=X
		..1.		PDFHLDD	"X'20'" ENTRY FOUND BY $$NQ DRY RUN
		...1		PDFLFC	"X'10'" PFLUSH/PCANCEL BY OPERATOR
		.... 1...		PDFLFD	"X'08'" PFLUSH PARTLY DONE BY $$XR
		.... .1..		PDFSGMNT	"X'04'" ..SEGMENT REQUEST
		.... ..1.		PDFXWUP	"X'02'" DYN. EXEC. WRITER IS UP NOW
(38)	56	ADDRESS	4	PDJH	PTR TO JOB HEADER RECORD
(3C)	60	ADDRESS	4	PDJT	PTR TO JOB TRAILER RECORD

Statistical Information.
This information is destined for the
execution account record and there is a pointer to the SLI work area.

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(40)	64	ADDRESS	4	PDSL	PTR TO SLI WORKAREA
(44)	68	SIGNED	4	PD#L	NR OF LINES SPOOLED
(48)	72	SIGNED	4	PD#C	NR OF CARDS SPOOLED
(4C)	76	SIGNED	2	PD#P	NR OF PAGES SPOOLED
(4E)	78	CHARACTER	1	PDOC	DEFAULT OUTPUT CLASS
(4F)	79	BITSTRING	1	PDMT	MULTTASK INDICATOR
		CHARACTER		PDMTI	"C'M'" .. MULTI-TASK PARTITION ID
(50)	80	CHARACTER	4	PDMRC	MAX. RETURN CODE

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(54)	84	CHAR-ACTER	4	PDLRC	LAST RETURN CODE
(58)	88	SIGNED	4	PD#PG	NR OF PAGES SPOOLED TOTALLY
(5C)	92	BITSTRING	1	PDFLG2	FLAG BYTE 2
		1...		PDFHWFW	"X'80'" ..WFW MESSAGE DESIRED
(5D)	93	BITSTRING	1	PDFLG7	JCL FLAG 7
(5E)	94	BITSTRING	1	PDFLG8	JCL FLAG 8
(5F)	95	BITSTRING	1		RES. FOR FUTURE

3540 Spool Device Entry (same format as RDR device entry)

THE FOLLOWING TWO SIXTEEN BYTES FIELDS ARE USED
TO GENERATE 3540 DEVICE LIST ENTRIES WHEN 3540
DATA IS TO BE SPOOLED. THE FORMAT OF THE ENTRY
IS DESCRIBED BY THE IPW$DDE MACRO INSTRUCTION.

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(60)	96	SIGNED	4	PDER1 (4)	FIRST 3540 SPOOL ENTRY
(70)	112	SIGNED	4	PDER2 (4)	SECOND 3540 SPOOL ENTRY
		1...		PDLN	"*-PDDS" BASIC LENGTH OF CTRL BLOCK
		1		PDMAXSR	"1" MAX NUMBER OF SPOOLED RDR
		.... 111.		PDMAXSO	"14" MAX NUMBER OF SPOOLED OUT

THE REMAINDER OF THE BLOCK CONTAINS THE PARTITION
DEVICE LIST, CONSISTING OF ONE OR MORE SIXTEEN-BYTE
ENTRIES, EACH ONE OF WHICH DESCRIBES ONE OF THE
SPOOL DEVICES ASSOCIATED WITH THE PARTITION. THE
FORMAT OF EACH ENTRY IS DESCRIBED BY THE IPW$DDE
MACRO INSTRUCTION.
THERE CAN EXIST AT MOST 30 ENTRIES:
 AT MOST 1 RDR ENTRY
 AT MOST 14 LST ENTRIES
 AT MOST 14 PUN ENTRIES
 1 DUMMY ENTRY AS END OF LIST INDICATOR
THE FOLLOWING STATEMENTS PROVIDE ADDITIONAL
DESCRIPTION OF THE FIRST OF THESE ENTRIES - THAT
FOR THE PARTITION READER DEVICE.

```
Bytes       Label
Hex.        of Field    Description/Function
----------------------------------------------------------------------
•  RDR Device Entry (maximum = 1)

80-83       PDPU        Address of entry in the VSE/AF
                        PUB for a card reader device
84-87       PDTC        Address of execution reader TCB
88-8B       PDCB        CCB address.
                        The first byte of this field is the SVC code:
                        X'00'=SVC  0: I/O request by user program
                        X'90'=SVC 90: accounting request by PA
                        X'91'=SVC 91: accounting request by JCL
8C          PDDT        Device type code
8D          PDCL        Device class code
                        can be R = normal reader,
                        or C = console
8E-8F       PDRQ        Requestor ID

•  LST Device Entry (maximum = 14) (Definition Macro IPW$DDE)

00-03       TLPU        Address of entry in the VSE/AF
                        PUB for a printer device
04-07       TLTC        Address of the execution list TCB
                        .. X'04" - Identify IPWSEGM Request
08-0B       TLCB        CCB address
0C          TLDT        Device type code
0D          TLCL        For list device entry this can be
                        L = device is being spooled,
                        N = device is not being spooled.
0E-0F       TLRQ        Requestor ID

 •  PUN Device Entry (maximum = 14).  Same format as LST device entry.
    The addresses depend on the number of LST entries.
    (Definition Macro IPW$DDE)

            TLPU        Address of entry in the VSE/Advanced Functions
                        PUB for a punch device
            TLTC        Address of the execution punch TCB
            TLCB        CCB address
            TLDT        Device type code
            TLCL        For punch device entry this can be
                        P = device is being spooled,
                        N = device is not being spooled.
            TLRQ        Requestor ID
```

How to Locate: Refer to Figure 152 on page 731 in Chapter 6, “Diagnostic Aids.”

Physical Data Record Area (PDA)

Space for this area is reserved during the execution of a physical reader/writer routine. The size of the area depends on the specifications in the DBLK parameter. It consists of a CCB and a CCW string which constitutes the channel program, followed by areas that contain the input or output data records.

Note: For an RJE task the CCB and the channel program is in the LCB. During a read operation the area is initialized by calculating the amount of data records and their CCWs that will fit in the area. Then an SVC 0 is issued to commence the I/O operation to read cards or 80 byte records into it. When it is full, the data is transferred to the logical data area by the function IPW$PLR and is ready for output to the spooling device assigned as the data file. Queue records are constructed on the queue file to record the seek addresses of the data on the data file.

During a write operation, the reverse occurs. Data is read from the spooling device to the LDA from where it is transferred to this PDA ready for the physical routine to print or punch the data.

How to Locate: Refer to Figure 152 on page 731 in Chapter 6, “Diagnostic Aids.”

Physical Work Space (PWS)

Definition Macro: IPW$DPW

The physical work space is used to address and save the information necessary for reentrance of the physical reader/writer. The area for PWS is reserved by the physical routine. It records information that points to a physical data area.

Note: There is no PWS for an RJE task; it is replaced by information contained in the LCB or SUCB/LUCB, respectively.

Bytes	Label	
Hex.	of Field	Description/Function
00-03	PBV1	Virtual address of the first PDA
04-07	PBR1	Real address of the first PDA
08-0B	PBV2	Virtual address of the second PDA
0C-0F	PBR2	Real address of the second PDA
10-13	PWVE	Virtual address of the active PDA
14-17	PWRE	Real address of the active PDA
18-19	PWLC	Displacement of last CCW in string from beginning of PDA
1A-1B	PWRL	Physical record length: to update the record pointer in the deblock routine
1C-1F	PWDI	Device type information
	PWDB	X'02' - Double buffer indicator
1D	PWDT	Device type of unit record device
1E-1F	PWLU	LUB number
20-23	PWDV	Virtual address of end of PDA
24-27	PWDA	Real address of end of PDA
28-2B	PWCA	Real address of the first CCW
2C	PWOT	Operation byte
	PWWC	X'80' - wait for completion request
2D	PWML	Message reply length
2E-35	PWRA	Message reply area
36-37	PWFS	Standard FCB name suffix

3540 Physical Work Space

Definition Macro: IPW$DPW E3540=YES

The 3540 physical work space is used to address and save the information necessary for diskette processing. The work space is either reserved by the physical routine (IPW$$PR) in case of alternate diskette processing, by the process diskette record routine (IPW$$ER) in case of primary diskette processing or by the logical reader routine (IPW$$LR) for dynamic * $$ RDR processing. The address of the 3540 physical work space currently in use is stored in the TCB field 'TC3W'.

Bytes	Label	
Hex.	of Field	Description/Function
00-03	PERA	Real address of the physical work space
04-07	PEDI	Device type indication.
04		Reserved.
05	PEDT	Device type.
06-07	PELU	Programmer logical unit.
08-0B	PECU	device address of diskette unit ('cuu')
0C-0F	PEHA	address of higher level 3540 PWS
10-1F	PEDP	Diskette parameters from PSTART.
10-17	PEFI	File identification.
18-1B	PEPS	PSTART parameters.
18	PEOP	Option byte feed for 3540.
	PEFD	X'01' - Feed 3540
19	PEND	Number of diskettes to be read.
1A	PESC	Sequence check required.
1B	PEVE	Verify requested.
1C-1F	PECD	Displacement between real and virtual CCB addresses.
20-23	PECV	Address of 3540 CCB or physical data area
24-27	PEDV	Virtual address of first 3540 data buffer
28-2B	PEDA	Real address of first 3540 data buffer
2C-2F	PEVN	Virtual address of second data buffer.
30-33	PERN	Real address of second data buffer.
34-37	PEBS	Real address of forced pre-SEEK CCW.
38-3B	PESK	Seek address (00CCHHRR).
3C-3F	PESO	Overlap seek address (00CCHHRR).
40-43	PELO	Extent lower limit (00CCHHRR).
44-47	PEED	Next sector address (00CCHHRR).
48-49	PERL	Record length.
4A-4B	PENN	No. of buffers allocated in 2nd data buffer.
4C-4D	PESI	Sequence identification.
4C	PEMI	Multi-volume identification.
4D	PESN	Volume sequence number.
4E	PEOC	Open return code.
4F	PEOD	Number of opened diskettes.
50-57	PEDW	Double word for conversion purposes.

How to Locate: Refer to Figure 152 on page 731 in Chapter 6, “Diagnostic Aids.”

PNET Control Block (PNCB)

Definition Macro: IPW$DPN

This macro is used to define the master control block for the PNET function. The PNCB is created at initialization time if the PNET parameter is specified at VSE/POWER generation time. Its address can be found in the CAT at label 'CAPN'.

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
PNET MASTER CONTROL BLOCK					
THE PNET CONTROL BLOCK (PNCB) IS CREATED AT VSE/POWER INITIALIZATION, IF THE PNET FEATURE IS INSTALLED. THE POINTER TO THE PNCB IS ANCHORED IN THE CAT LABEL 'CAPN' NETWORK DEFINITION TABLE HEADER					
(0)	0	STRUCTURE	0	PNCBDS	
(0)	0	CHARACTER	16	PNCBSD	SECTION DESCRIPTOR
(10)	16	ADDRESS	4	PNCBNCB	ADDR. OF FIRST NCB
(14)	20	ADDRESS	4	PNCBTLD	ADDRESS OF LINE DRIVER TCB
(18)	24	ADDRESS	4	PNCBNDT	ADDR. OF NETWORK DEFINITION TABLE
(1C)	28	ADDRESS	4	PNCBLKW	RESERVED FOR LOCKWORD
(20)	32	ADDRESS	4	PNCBVDCB	ADDR. OF SNA CONTROL BLOCK
(24)	36	ADDRESS	4	PNCBTNT	PTR TEMPORARY NAT TABLE
(28)	40	ADDRESS	4	PNCBTDCB	ADDR. OF TD-SUBTASK C-BLOCK
(2C)	44	ADDRESS	4	PNCBSDCB	ADDR. OF SD-SUBTASK C-BLOCK
(30)	48	ADDRESS	4	PNCBSTCB	CNSLTR SYSLST TRACE CCB ADDR
(34)	52	BITSTRING	1	PNCBTRTA	FLAG - SYSLST TRACE ACT(LOCK
(35)	53	BITSTRING	1	PNCBSTDT	SYSLST PUB DEVICE TYPE
(36)	54	BITSTRING	1	PNCBTRST	FLAG - SYSLST ASSIGNED
(37)	55	BITSTRING	1	PNCBTRSI	FLAG - SYSLST INITIALIZED
(38)	56	ADDRESS	4		RESERVED FOR FUTURE USE
(3C)	60	ADDRESS	4		RESERVED FOR FUTURE USE
LOAD ADDRESS LIST FOR PNET PHASES					
(40)	64	CHARACTER	4	PNCBALD	LINE DRIVER IPW$$LD
(44)	68	CHARACTER	4	PNCBALD1	LINE DRIVER IPW$$LD1
(48)	72	CHARACTER	4	PNCBALD2	LINE DRIVER IPW$$LD2
(4C)	76	CHARACTER	4	PNCBALD3	LINE DRIVER IPW$$LD3
(50)	80	CHARACTER	4	PNCBALD4	LINE DRIVER IPW$$LD4
(54)	84	CHARACTER	4	PNCBALD5	LINE DRIVER IPW$$LD5
(58)	88	CHARACTER	4	PNCBANM	I/O MANAGER
(5C)	92	CHARACTER	4	PNCBANR	RECEIVER
(60)	96	CHARACTER	4	PNCBANR2	RECEIVER PART 2
(64)	100	CHARACTER	4	PNCBANP	PRESENTATION SERVICE
(68)	104	CHARACTER	4	PNCBANT	TRANSMITTER
(6C)	108	CHARACTER	4	PNCBANC	COMPOSER
(70)	112	CHARACTER	4	PNCBANK	COMPRESSION ROUTINE

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(74)	116	CHARACTER	4	PNCBAOP	VTAM SUBTASK ENTRY POINT
(78)	120	CHARACTER	4	PNCBASE	SNA EXIT ROUTINES
(7C)	124	CHARACTER	4	PNCBASR	SNA SEND/RECEIVE FUNCTION AND EXIT
(80)	128	CHARACTER	4	PNCBACT	SNA SESSION BUILD (CONNECT)
(84)	132	CHARACTER	4	PNCBADT	SNA SESSION TERMINATE (DISCONNECT)
(88)	136	CHARACTER	4	PNCBABS	PHASE ADDR. OF BUFFER SERVICE
(8C)	140	CHARACTER	4	PNCBATD	PHASE ADDR.(TCP/IP DRIVER)
(90)	144	CHARACTER	4	PNCBATS	PHASE ADDR.(IP SOCKET PROC.)
(94)	148	CHARACTER	4	PNCBASD	PHASE ADDR.(SSL DRIVER)
(98)	152	CHARACTER	4	PNCBASS	PHASE ADDR.(SSL SOCKET PROC)
(9C)	156	CHARACTER	4	PNCBCPS	PHASE ADDR. OF PSTART CMD PROCESSOR
(A0)	160	CHARACTER	4	PNCBCPF	PHASE ADDR. OF PFLUSH CMD PROCESSOR
		...1 1..1		PNCBHNR	"(*-PNCBALD)/4" NUMBER OF PHASES
THE FOLLOWING TWO ADDRESSES CONTAIN THE POINTER TO THE ERROR EXIT ROUTINES FOR BOTH THE TRANSMITTER AND RECEIVER.					
(A4)	164	ADDRESS	4	PNCBERNR	ERROR EXIT ROUTINE OF RECEIVER
(A8)	168	ADDRESS	4	PNCBERNT	ERROR EXIT ROUTINE OF TRANSMITTER
(AC)	172	ADDRESS	4	PNCBAUE	USER READER EXIT
(B0)	176	SIGNED	2	PNCBAUEL	PNET EXIT WOR AREA SIZE
(B2)	178	SIGNED	2	PNCBTUEL	LENGTH OF XMTEXIT WORK AREA
(B4)	180	ADDRESS	4	PNCBTUE	ADDRESS OF XMTEXIT
(B8)	184	BITSTRING	2		RESERVED FOR FUTURE USE
(BA)	186	BITSTRING	1	PNCBST1	STATUS BYTE ONE
		1...		PNCB1DPS	"X'80'" .. DELAY PSTART PNET FOR SNA
(BB)	187	BITSTRING	3		RESERVED
OWN NODE INFORMATION					
(BE)	190	CHARACTER	8	PNCBONN	NODE NAME OF OUR SYSTEM
(C6)	198	CHARACTER	1	PNCBONQ	.. QUALIFIER
(C7)	199	CHARACTER	8	PNCBNDTN	NDT PHASE NAME
(CF)	207	CHARACTER	9	PNCBAPPL (0)	OUR NODE'S APPL-ID PARAMETER
(CF)	207	BITSTRING	1	PNCBALTH	LENGTH OF APPLID (PNET USES DEFAULT OF 8)
(D0)	208	CHARACTER	8	PNCBAPID	OUR NODE'S APPL-ID
(D8)	216	ADDRESS	4	PNCBSECB	SUBTASK-ECB FOR VTAM
(DC)	220	ADDRESS	4		RESERVED FOR FUTURE USE
DC (((-&PR.SD-1+8)/32+1) 32-(-&PR.SD+8))AL1(0)					
		111.		PNCBLN	"*-PNCBSD" LENGTH OF CONTROL BLOCK

PNET TCP Driver Control Block (TDCB) and PNET SSL Driver Control Block (SDCB)

Definition Macro: IPW$DTP

This macro is used to define the master control block of the PNET TCP/IP interface (represented by the TD-Subtask) for the PNET TCP function and the PNET SSL interface (represented by the SD-Subtask) for the PNET SSL function. The TDCB/SDCB is created at ititialization time immediately after the PNCB creation. Its address can be found in the PNCB at label 'PNCBTDCB' respectively 'PNCBSDCB'. Both control blocks are created by the same macro.

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
					TDCP CONTROL BLOCK
(0)	0	STRUCTURE	0	TDCBDS	START OF DSECT
(0)	0	CHARACTER	16	TDCBSD	SECTION DESCRIPTOR
(10)	16	ADDRESS	4	TDCBECB	PNET TCP/IP EVENT CONTROL BLOCK
(14)	20	SIGNED	4	TDCBGTIV	GENRAL TIMER INTERVAL
(18)	24	ADDRESS	4	TDCBNDTE	POINTER TO LOCAL NDT ENTRY
(1C)	28	BITSTRING	24	TDCBTIME	RESERVED FOR TIMER ELEMENT
(34)	52	ADDRESS	4	TDCBSECB	TD-SUBTASK ALIVE(0) OR DOWN(1) ECB
(38)	56	ADDRESS	4	TDCBATCB	POINTER TO 'PSTART TCPIP' TASK TDCB POINTER TO 'PSTART TCPSSL' TASK SDCB
TDCB GENERAL CONTROL FIELDS					
(3C)	60	BITSTRING	8	TDCBTRC1 (0)	START OF TRACED INFO
(3C)	60	SIGNED	2	TDCBNONU	NUMBER OF STARTED TCP/IP NODES
(3E)	62	BITSTRING	1	TDCBACT1	TDCB DETACH ACTIVITY FLAG BYTE 1
		.1..		TDCBA1TI	"X'40'" .. TERMINATE IMMEDIATELY (ERROR)
		..1.		TDCBA1TE	"X'20'" .. TERMINATE AT EOJ (PEND)
		.... 1...		TDCBA1PE	"X'08'" .. PSTOP TCP/IP AT EOJ
		.... .1..		TDCBA1PI	"X'04'" .. PSTOP TCP/IP IMM (NOT YET USED)
		.... ..1.		TDCBA1DT	"X'02'" .. DETACH TCP/IP SUBTASK
(3F)	63	BITSTRING	1	TDCBACT2	TDCB ACTIVITY FLAG BYTE 2
		1...		TDCBA2CL	"X'80'" .. CLOSE PASSIVE MODE
		.1..		TDCBA2TR	"X'40'" .. TRACE SOCKETCALL OF PASSIVE MODE
		.... 1...		TDCBA2AF	"X'08'" .. ACCEPT FAILED ONCE
		.... .1..		TDCBA2AX	"X'04'" .. ACCEPT FAILED TWICE IN A ROW
		.... ..1.		TDCBA2RP	"X'02'" .. RESTART PASSIVE CONN
(40)	64	BITSTRING	1	TDCBACT3	TDCB ACTIVITY FLAG BYTE 3
		...1		TDCBA3PL	"X'10'" .. POST PNET LINE DRIVER
		.... 1...		TDCBA3IV	"X'08'" .. AT LEAST 1 TQE-ECB POSTED
(41)	65	BITSTRING	1	TDCBSTA1	TCP/IP STATUS BYTE
		1...		TDCBS1IA	"X'80'" .. API INTERFACE AVAILABLE SET IF 1ST SOCKETCALL SOCKET OK RESET IF TERMAPI ISSUED
		.1..		TDCBS1DP	"X'40'" .. SUBTASK IN $$AT BEFORE DETACH
		..1.		TDCBS1SM	"X'20'" .. ISSUE START-UP MSG ON CONSOLE
		...1		TDCBS1RC	"X'10'" .. RECURSIVE FLAG: BEEN IN $$AT
		.... 1...		TDCBS1PS	"X'08'" .. PASSIVE SOCKET NOT USABLE
		.... .1..		TDCBS1IT	"X'04'" .. SOK INITAPI SUCESSFUL
		.... ..1.		TDCBS1P1	"X'02'" .. PASSIVE SOCKET ONCE SUCC
(42)	66	BITSTRING	1	TDCBSTA2	TCP/IP STATUS BYTE
		1...		TDCBS2NW	"X'80'" .. NO WAIT WITHIN MAINLINE
(43)	67	BITSTRING	1	TDCBTTC	TCP/IP-TERMINATION CODE
		1...		TDCBTTCV	"X'80'" .. TCP/IP TERMINATED
		.... 1...		TDCBTRE1	"*-TDCBTRC1" LENGTH OF TRACED INFO
(44)	68	BITSTRING	1	TDCBTTCQ	TERMINATION QUALIFIER
(45)	69	BITSTRING	1	TDCBRCNT	RETRY COUNTER
(46)	70	BITSTRING	1	TDCBPROC	PROCESS BYTE
		...1		TDCBTRCE	"X'10'" .. TRACE SOCKETCALL

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(47)	71	CHARACTER	1		FILLER FOR ALIGNMENT
FIELDS USED BY SD-SUBTASK					
(48)	72	BITSTRING	1	TDCBSLS1	TCP/IP STATUS BYTE
		1...		TDCBSL1I	"X'80'" .. SSL INTERFACE AVAILABLE
(49)	73	CHARACTER	1		RESERVED FOR FUTURE USE
(4A)	74	CHARACTER	1		RESERVED FOR FUTURE USE
(4B)	75	CHARACTER	1		RESERVED FOR FUTURE USE
(4C)	76	SIGNED	2		RESERVED FOR FUTURE USE
(4E)	78	CHARACTER	1		RESERVED FOR FUTURE USE
(4F)	79	CHARACTER	8	TDCBSECT	SECURITY PROTOCOL
(57)	87	BITSTRING	1		END DELIMITER FOR SEC PROT
(58)	88	CHARACTER	16	TDCBKEYR	LIB.SUBLIB NAME
(68)	104	BITSTRING	1		END DELIMITER FOR KEYRING
(69)	105	CHARACTER	1		RESERVED FOR FUTURE USE
(6A)	106	SIGNED	2		RESERVED FOR FUTURE USE
(6C)	108	SIGNED	4	TDCBTOUT	TIMEOUT FOR SSL
(70)	112	SIGNED	4	TDCBSXRN	NUMBER OF SOK DESCRIPTORS USED BY SELECTEX FOR READ
(74)	116	SIGNED	4	TDCBSXWN	NUMBER OF SOK DESCRIPTORS USED BY SELECTEX FOR WRITE
(78)	120	DBL WORD	8	TDCBSXTI	TIME TO WAIT TILL POSTED USED BY SELECTEX FOR WRITE
AREAS USED FOR SOCKET CALL : INITAPI					
(80)	128	SIGNED	2	TDCBSCAT	API TYPE
(82)	130	SIGNED	2	TDCBSCMX	MAXIMUM NUMBER OF SOCKETS
(84)	132	SIGNED	4	TDCBSCMN	MAXIMUM DESCRIPTOR NUMBER
(88)	136	CHARACTER	8	TDCBSCSI	SUBTASK IDENTIFIER
(90)	144	ADDRESS	4		RESERVED FOR FUTURE USE
(94)	148	ADDRESS	4	TDCBSCIP	IP-ADDRESS IN BINARY FORMAT
(98)	152	CHARACTER	15	TDCBSCIR	IP-ADDRESS IN READABLE FORMAT
(A7)	167	CHARACTER	1		FILLER FOR ALIGNMENT
(A8)	168	BITSTRING	24	TDCBPTQE (0)	TQE-ELEMENT: TIME LIMIT AFTER WHICH PASSIVE CONNECTION COMPLETED
(A8)	168	BITSTRING	12		NOT REFERENCED
(B4)	180	BITSTRING	4	TDCBPTQP	ECB TO BE POSTED
(B8)	184	BITSTRING	5		NOT REFERENCED
(BD)	189	CHARACTER	3	TDCBPTQY	EYE-CATCHER
(C0)	192	ADDRESS	4	TDCBATDY	ADDRESS OF TIDY-UP ROUTINE OF .. .. IPW$$TD, RESP. IPW$$SD
(C4)	196	SIGNED	2	TDCBSTIK	TIK OF SUBTASK
(C6)	198	BITSTRING	1		RESERVED FOR FUTURE USE
(C7)	199	BITSTRING	1		RESERVED
(C8)	200	ADDRESS	4	TDCBTQEA	ANCHOR OF TQE CHAIN
(CC)	204	ADDRESS	4		RESERVED
(D0)	208	ADDRESS	4	TDCBMSGD	DOM-ID OF MSG
(D4)	212	BITSTRING	1		RESERVED
(D5)	213	BITSTRING	1		RESERVED
(D6)	214	BITSTRING	1		RESERVED
(D7)	215	BITSTRING	1		RESERVED FOR FUTURE USE
(D8)	216	ADDRESS	4	TDCBTPER	ERROR TIME USED FOR RESTART
(DC)	220	ADDRESS	4	TDCBAMRT	RESERVED FOR FUTURE USE

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(E0)	224	SIGNED	4	TDCBFION	PARA FOR IOCTL QOCKET CALL
(E4)	228	ADDRESS	4		RESERVED FOR FUTURE USE
(E8)	232	ADDRESS	4		RESERVED FOR FUTURE USE
(EC)	236	ADDRESS	4		RESERVED FOR FUTURE USE
(F0)	240	ADDRESS	4		RESERVED FOR FUTURE USE
(F4)	244	ADDRESS	4		RESERVED FOR FUTURE USE
(F8)	248	ADDRESS	4		RESERVED FOR FUTURE USE
(FC)	252	ADDRESS	4		RESERVED FOR FUTURE USE
AREAS USED FOR SOCKET CALL : LISTEN BACKLOG NUMBER = MAXIMUM NUMBER OF SOCKETS USED IN SOCKETCALL INITAPI (TDCBSCMX) TCP/IP WORKAREA USED WITHIN NCB AND TDCB					
(100)	256	SIGNED	2	TDNTPDS (0)	
(100)	256	CHARACTER	15	TDNTPIPC	IP-ADDR IN READABLE FORMAT
(10F)	271	CHARACTER	1	TDNTPTYP	TYPE OF ITP WORKAREA
		11.. ...1		TDNTPTYA	"C'A'" .. A = ACTIVE = NCB
		11.1 .111		TDNTPTYT	"C'P'" .. P = PASSIVE = TDCB
START OF AREA-1 TO BE TRACED					
(110)	272	BITSTRING	36	TDNTPTC1 (0)	START OF TRACED INFO 1
(110)	272	BITSTRING	1	TDNTPST1	TCP/IP STATUS BYTE 1:
GENERAL TCP/IP STATUS, SOCKETCALL STATUS					
		1...		TDNTPS1T	"X'80'" .. TCP/IP INIT CONTACT COMPL
		.1..		TDNTPS1F	"X'40'" .. TCP/IP CONN. CLOSED
		..1.		TDNTPS1R	"X'20'" .. TCP/IP RESTART: NAK-3
		...1		TDNTPS1E	"X'10'" .. TCP/IP LINE ERROR
		.... 1...		TDNTPS1A	"X'08'" .. PROCESSING ACTIVE MODE
		.... .1..		TDNTPS1I	"X'04'" .. 1.SOCKETCALL ISSUED
		.... ..1.		TDNTPS1L	"X'02'" .. SSL FEATURE INITIATED
		1		TDNTPS1S	"X'01'" .. STOP CONNECTION
(111)	273	BITSTRING	1	TDNTPST2	TCP/IP STATUS BYTE 2
GENERAL NODE STATUS					
		1...		TDNTPS2I	"X'80'" .. CTC I/O ONCE PROCESSED
		.1..		TDNTPS2R	"X'40'" .. RESTART TCP/IP PV010222
		..1.		TDNTPS2B	"X'20'" .. FIRST COMES TTB
		...1		TDNTPS2C	"X'10'" .. CLOSE CONNECTION
		.... 1...		TDNTPS2O	"X'08'" .. OPEN-CTRL-REC. RECEIVED
		.... .1..		TDNTPS2A	"X'04'" .. ACK-CTRL-REC. SENT
		.... ..1.		TDNTPS22	"X'02'" .. NAK WITH RC=2 SENT
		1		TDNTPS2W	"X'01'" .. WAIT THAT REMOTE ISSUES .. .. CONNECT
(112)	274	BITSTRING	1	TDNTPST3	TCP/IP STATUS BYTE 3
STATUS: CTC I/O					
		1...		TDNTPS3S	"X'80'" .. CTC I/O STARTED
		.1..		TDNTPS3C	"X'40'" .. CTC I/O TO BE COMPLETED
		..1.		TDNTPS3Z	"X'20'" .. CCW-WRITE DATA SENT
		...1		TDNTPS3Y	"X'10'" .. CCW-READ DATA RECVED
		.... 1...		TDNTPS3B	"X'08'" .. CTC I/O WITHOUT BUFFER
		.... .1..		TDNTPS3N	"X'04'" .. TCP BLOCK PARTLY RCVED
		.... ..1.		TDNTPS3L	"X'02'" .. LEAVE IDLING STATE
		1		TDNTPS3I	"X'01'" .. IDLING(NOTHING SENT/RCV)
(113)	275	BITSTRING	1	TDNTPST4	TCP/IP STATUS BYTE 4
STATUS: MISCELLANEOUS					
		1...		TDNTPS4P	"X'80'" .. WAIT FOR POSTED ECB
			.1..		TDNTPS4C	"X'40'" .. CONNECTION CLOSED
		..1.		TDNTPS4T	"X'20'" .. TERMINATE LINE
		...1		TDNTPS4W	"X'10'" .. WAIT TILL TIME EXPIRED
		.... 1...		TDNTPS4A	"X'08'" .. CANCEL ISSUED
		.... .1..		TDNTPS4L	"X'04'" .. CLOSE ISSUED

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
		.... ..1.		TDNTPS4N	"X'02'" .. SOK NUMBER TOO HIGH (SSL)
(114)	276	BITSTRING	1	TDNTPST5	TCP/IP STATUS BYTE 5:
CLOSING CODES, CLOSED DUE TO:					
		1...		TDNTPS5U	"X'80'" .. SIGNOFF REC. SEND
		.1..		TDNTPS5R	"X'40'" .. SIGNOFF REC. RECEIVED
		..1.		TDNTPS5A	"X'20'" .. INVALID DEFINITION
		...1		TDNTPS5N	"X'10'" .. TCP NJE NAK RECEIVED
		.... 1...		TDNTPS58	"X'08'" .. CAUSED BY TCP/IP RC=8
		.... .1..		TDNTPS5C	"X'04'" .. CAUSED BY TCP/IP RC=12
		.... ..1.		TDNTPS5I	"X'02'" .. POWER INTERNAL ERROR
		1		TDNTPS5S	"X'01'" .. CAUSED BY REMOTE CLOSED
(115)	277	BITSTRING	1	TDNTPST6	TCP/IP STATUS BYTE 6:
		1...		TDNTPS6T	"X'80'" .. TRACE SOCKETCALL
		.1..		TDNTPS6I	"X'40'" .. INIT TIME INTERVAL
		1111 111.		TDNTPPEV	"X'FE'" .. SPECIAL EVENT HUSTEST
		....		TDNTPS6Y	"X'00'" .. SPECIAL RETRY HUSTEST
(116)	278	BITSTRING	1	TDNTPRV1	RESERVED
(117)	279	BITSTRING	1	TDNTPRV2	RESERVED
RETURN ADDRESSES FOR SOCKETCALL ROUTINE					
(118)	280	ADDRESS	4	TDNTPSO0	SOCKETCALL SUCCESSFUL
(11C)	284	ADDRESS	4	TDNTPSO4	SOCKETCALL SHOULD BE RETRIED
(120)	288	ADDRESS	4	TDNTPSO8	CONNECTION TO BE STOPPED
(124)	292	ADDRESS	4	TDNTPSOC	INTERFACE TO BE TERMINATED
INTERFACE AREA BETWEEN IPW$$TD AND IPW$$TS RESPECTIVELY BETWEEN IPW$$SD AND IPW$$SS					
(128)	296	SIGNED	4	TDNTPR1	RETURN CODE FROM $TS
		....		TDNTPR10	"0" .. OK
		.... .1..		TDNTPR14	"4" .. RETRY NECESSARY/POSSIBLE
		.... 1...		TDNTPR18	"8" .. TERMINATE CONNECTION
		.... 11..		TDNTPR1C	"12" .. TERMINATE INTERFACE
		.... ..1.		TDNTPR1R	"2" .. RETRY DUE TO IPW$$SD
(12C)	300	BITSTRING	1	TDNTPSC	SOCKETCALL REQUESTED
		1		TDNTPIA	"1" .. INITAPI
		.... ..1.		TDNTPTA	"2" .. TERMAPI
		11		TDNTPGL	"3" .. GETHOSTID
		.... .1..		TDNTPLI	"4" .. LISTEN
		.... .1.1		TDNTPAC	"5" .. ACCEPT
		.... .11.		TDNTPSD	"6" .. SEND
		.... .111		TDNTPRV	"7" .. RECEIVE
		.... 1...		TDNTPCL	"8" .. CLOSE
		.... 1..1		TDNTPCN	"9" .. CANCEL
		.... 1.1.		TDNTPGA	"10" .. GETHOSTBYADDR
		.... 1.11		TDNTPGN	"11" .. GETHOSTBYNAME
		.... 11..		TDNTPGS	"12" .. GET SOCKET
		.... 11.1		TDNTPBI	"13" .. BIND
		.... 111.		TDNTPCO	"14" .. CONNECT
		 1111		TDNTPSR	"15" .. SELECT USING READ-ARRAY
		...1		TDNTPSW	"16" .. SELECT USING WRITE-ARRAY
SOME SSL SOCKET CALLS:					
		...1 ...1		TDNSSLIN	"17" .. SSL INITIALIZE
		...1 ..1.		TDNSSLUN	"18" .. SSL UNINITIALIZE
		...1 ..11		TDNSSLGN	"19" .. SSL GET DNAME IN DB
		...1 .1..		TDNSSLFM	"20" .. SSL FREE MEMORY
		...1 .1.1		TDNSSLSI	"21" .. SSL SOCKET INITIALIZE
		...1 .11.		TDNSSLSR	"22" .. SSL SOCKET READ
		...1 .111		TDNSSLSW	"23" .. SSL SOCKET WRITE
		...1 1...		TDNSSLSC	"24" .. SSL SOCKET CLOSE
		...1 1..1		TDNSSLRS	"25" .. SSL SOCKET RESET
		...1 1.1.		TDNSSLGC	"26" .. SSL GET CIPHER INFO
		...1 1.11		TDNTPIOC	"27" .. IOCTL=SET NONBLOCKING
(12D)	301	BITSTRING	1		UNUSED

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(12E)	302	BITSTRING	2	TDNTPRYC	RETRY COUNTER
(130)	304	SIGNED	4	TDNTPTIV	TIMER INTERVAL TO BE SET
		..1. .1..		TDNTPTE1	"*-TDNTPTC1" LENGTH OF TRACED INFO 1
(134)	308	BITSTRING	24	TDNTPTQE (0)	TIMER QUEUE ELEMENT
(134)	308	BITSTRING	12		NOT REFERENCED
(140)	320	BITSTRING	4	TDNTPTEB	POST BYTES
(144)	324	BITSTRING	5		NOT REFERENCED
(149)	329	CHARACTER	3	TDNTPTQY	EYECATCHER
AREAS USED FOR SOCKET CALLS AREAS USED FOR SEVERAL SOCKET CALLS START OF AREA-2 TO BE TRACED					
(14C)	332	ADDRESS	4	(0)	ALIGN
(14C)	332	BITSTRING	20	TDNTPTC2 (0)	START OF TRACED INFO 2
(14C)	332	SIGNED	2	TDNSCSOD	SOCKET DESCRIPTOR
(14E)	334	SIGNED	2		RESERVED
AREAS USED FOR SOCKET CALLS: BIND, ACCEPT, CONNECT					
(150)	336	BITSTRING	16	TDNSCDNM (0)	
(150)	336	SIGNED	2	TDNSCBFM	ADDRESSING FAMILY
(152)	338	SIGNED	2	TDNSCBPT	PORT NUMBER
(154)	340	SIGNED	4	TDNSCDIP	IP-ADDRESS
(158)	344	BITSTRING	4		RESERVED FOR SOCKETCALL
(15C)	348	BITSTRING	4		RESERVED FOR SOCKETCALL
		...1 .1..		TDNTPTE2	"*-TDNTPTC2" LENGTH OF TRACED INFO 2
AREAS USED FOR SEVERAL SOCKET CALLS EXCEPT SEND AND CANCEL					
(160)	352	ADDRESS	4	(0)	ALIGN
(160)	352	BITSTRING	1	TDNSCST1	STATUS OF SOCKETCALL
		1...		TDNSCS1S	"X'80'" .. SOCKETCALL STARTED
		.1..		TDNSCS1B	"X'40'" .. NO BUFFER AVAILABLE
(161)	353	BITSTRING	1		RESERVED
(162)	354	SIGNED	2	TDNSCCNT	RETRY COUNTER
(164)	356	ADDRESS	4	TDNSCBUF	BUFFER FOR RECV
(168)	360	ADDRESS	4	TDNSCNBY	NO OF BYTES FOR RECV
(16C)	364	SIGNED	4	TDNSCRC	RETURN CODE FROM SOCKETCALL
(170)	368	SIGNED	4	TDNSCERN	ERROR NUMBER
(174)	372	BITSTRING	164	TDNSCDCB (0)	
(174)	372	ADDRESS	4	TDNSCECB (0)	ECB
(174)	372	BITSTRING	2		.. UNREFERENCED
(176)	374	BITSTRING	1	TDNSCECP	.. POSTED BYTE
		1...		TDNSCECI	"X'80'" .. POST BIT
(177)	375	BITSTRING	1		.. UNREFERENCED
(178)	376	BITSTRING	160	TDNSCRQ	WORKAREA FOR EZASMI
AREAS USED FOR SOCKET CALL : INITAPI, LISTEN SEE TDCB AREAS USED FOR SOCKET CALLS: GETHOSTBYADDR					
(218)	536	ADDRESS	4	TDNSCHST	ADDR. OF HOSTNAME STRUCTURE
AREAS USED FOR SOCKET CALL : GETHOSTBYNAME AND AREA ..SCHST, SEE SOCKET CALL GETHOSTBYADDR					
(21C)	540	SIGNED	4	TDNSCHNL	LENGTH OF HOSTNAME
(220)	544	ADDRESS	4	TDNSCHNM	ADDRESS OF HOSTNAME
AREAS USED FOR SOCKET CALL: SEND					
(224)	548	ADDRESS	4	(0)	ALIGN
(224)	548	BITSTRING	8	TDNTPTC3 (0)	START OF TRACED INFO 3
(224)	548	BITSTRING	1	TDNSCSS1	STATUS OF SOCKETCALL

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
		1...		TDNSCD1S	"X'80'" .. SOCKETCALL STARTED
		.1..		TDNSCD1B	"X'40'" .. NO BUFFER AVAILABLE
(225)	549	BITSTRING	1		RESERVED
(226)	550	SIGNED	2	TDNSCSCT	RETRY COUNTER
(228)	552	ADDRESS	4	TDNSCBUS	SOCKETCALL TO BE CANCELLED
(22C)	556	ADDRESS	4	TDNSCNBS	UNREFERENCED
(230)	560	SIGNED	4	TDNSCSAC	RETURN CODE
(234)	564	SIGNED	4	TDNSCSAE	ERROR NUMBER
		...1 .1..		TDNTPTE3	"*-TDNTPTC3" LENGTH OF TRACED INFO 3
(238)	568	BITSTRING	164	TDNSCSAL (0)	
(238)	568	ADDRESS	4	TDNSCSAB (0)	ECB
(238)	568	BITSTRING	2		.. UNREFERENCED
(23A)	570	BITSTRING	1	TDNSCSAP	.. POSTED BYTE
(23B)	571	BITSTRING	1		.. UNREFERENCED
(23C)	572	BITSTRING	160	TDNSCSAR	WORKAREA FOR EZASMI
AREAS USED FOR SOCKET CALL: CANCEL PARTLY RE-USED FOR SOCKET CALL: GET CIPHER INFO					
(2DC)	732	ADDRESS	4	(0)	ALIGN
(2DC)	732	BITSTRING	1	TDNSCCS1	STATUS OF SOCKETCALL
		1...		TDNSCC1S	"X'80'" .. SOK CALL STARTED, UNUSED
		.1..		TDNSCC1B	"X'40'" .. NO BFR AVAILABLE, UNUSED
(2DD)	733	BITSTRING	1		RESERVED
(2DE)	734	SIGNED	2	TDNSCCCT	RETRY COUNTER
(2E0)	736	ADDRESS	4	TDNSCBUC	SOCKETCALL TO BE CANCELLED
(2E4)	740	ADDRESS	4	TDNSCNBC	UNREFERENCED
(2E8)	744	SIGNED	4	TDNSCCAC	RETURN CODE
(2EC)	748	SIGNED	4	TDNSCCAE	ERROR NUMBER
(2F0)	752	BITSTRING	164	TDNSCCAL (0)	
(2F0)	752	ADDRESS	4	TDNSCCAB (0)	ECB
(2F0)	752	BITSTRING	2		.. UNREFERENCED
(2F2)	754	BITSTRING	1	TDNSCCAP	.. POSTED BYTE
(2F3)	755	BITSTRING	1		.. UNREFERENCED
(2F4)	756	BITSTRING	160	TDNSCCAR	WORKAREA FOR EZASMI
PARTLY RE-USED FOR SOCKET CALL: GET-CIPHER-INFO					
(2F0)	752	BITSTRING	104	TDNSSLCO (0)	OUTPUT OF GET-CIPHER-INFO
(2F0)	752	SIGNED	4		SYSTEM SSL VERSION
(2F4)	756	BITSTRING	64	TDNSSLCC	SPECS OF GET-CIPHER-INFO
(334)	820	BITSTRING	32		UNREFERENCED
(354)	852	BITSTRING	4		UNREFERENCED
(394)	916	SIGNED	4		RESERVED
NO SPECIAL AREAS FOR SOCKET CALLS: CLOSE, SOCKET, GETHOSTID, TERMAPI SEE AREAS USED FOR SEVERAL SOCKET CALLS AREAS USED FOR SSL SOCKET CALLS					
(398)	920	ADDRESS	4	TDNSSLCB	ADDR OF SSL CONTROL BLOCK RETURNED BY SSL-SOCK-INIT
(39C)	924	SIGNED	4	TDNSSLRC	REASON-CODE OF SOCK-INIT
(3A0)	928	ADDRESS	4	TDNSSLDN	ADDR OF DISTINGUISHED NAME RETURNED BY SSL-GETDNBYLAB INPUT FOR SSL-FREEMEM
(3A4)	932	ADDRESS	4	TDNSSLCF	ADDR OF CLIENT-CERTIFICATE UPDATED BY SSL-SOCK-INIT
(3A8)	936	ADDRESS	4	TDNSSLST	ADDR OF CLIENT-CERTIFICATE UPDATED BY SSL-SOCK-INIT
(3AC)	940	SIGNED	2	TDNSSLCS	2 BYTES OF SELECTED CIPHERS IS PART WITHIN ---SLCPO
(3AE)	942	SIGNED	2		RESERVED
(3B0)	944	SIGNED	4	TDNIOCMD	MODE FOR IOCTL
		....		TDNIOCBL	"0" .. BLOCKING MODE

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
		1		TDNIOCNB	"1" .. NONBLOCKING MODE
(3B4)	948	SIGNED	4		RESERVED
(3B8)	952	SIGNED	4		RESERVED
(3BC)	956	SIGNED	4		RESERVED
(3C0)	960	SIGNED	4		RESERVED
(3C4)	964	SIGNED	4		RESERVED
(3C8)	968	SIGNED	4		RESERVED
(3CC)	972	SIGNED	4		RESERVED
(3D0)	976	SIGNED	4		RESERVED
(3D4)	980	SIGNED	4		RESERVED
(3D8)	984	SIGNED	4		RESERVED
(3DC)	988	SIGNED	4		RESERVED
	WORKAREA FOR IPW$$TD, RESPECTIVELY IPW$$SD				
(3E0)	992	SIGNED	4	TDNTPNOB	NO OF BYTES SEND/RCVED
(3E4)	996	BITSTRING	33	TDNTPCTB	BUFFER FOR CTRL-RECORD
(405)	1029	BITSTRING	3	TDNTPCR1	RESERVED
(408)	1032	SIGNED	4	TDNTPBR1	BYTES RCVED VIA SOCKETCALL
(40C)	1036	SIGNED	4	TDNTPBP1	BYTES PROCESSED BY .. .. IPW$$TD, RESP. IPW$$SD
(410)	1040	ADDRESS	4	TDNTPWPO	ADDRESS: WAIT FOR POST ECB
(414)	1044	ADDRESS	4	TDNTPNCB	ADDRESS OF NCB
(418)	1048	BITSTRING	2		RESERVED
(41A)	1050	BITSTRING	2	TDNTPFCS	FCS SAVED FROM CTC I/O
(41C)	1052	BITSTRING	1		RESERVED
(41D)	1053	BITSTRING	1	TDNTPBCS	BCB SAVED FROM CTC I/O
(41E)	1054	BITSTRING	1	TDNTPBCI	BCB FOR INCOMING BUFFER
(41F)	1055	BITSTRING	1	TDNTPBCO	BCB FOR OUTGOING BUFFER
(420)	1056	SIGNED	4		RESERVED
(424)	1060	SIGNED	4		RESERVED
(428)	1064	SIGNED	4		RESERVED
(42C)	1068	SIGNED	4		RESERVED
(430)	1072	SIGNED	4		RESERVED
(434)	1076	SIGNED	4		RESERVED
(438)	1080	SIGNED	4		RESERVED
(43C)	1084	SIGNED	4		RESERVED
(43C)	1084		0	TDNTPLST	"*" END OF WORKAREA
(43C)	1084		0	TDNTPLN	"*-TDNTPDS" LENGTH OF WORKAREA
	AREA NOT TO BE CLEARED AFTER SOCKET CALL CLOSE				
(440)	1088	SIGNED	4	TDNSSLDS (0)	' START OF SSL WORKAREA
(440)	1088	CHAR-ACTER	8	TDNSSLKY	MEMBER IN SUBLIB
(448)	1096	BITSTRING	1		END DELIMITER FOR DNAME
(449)	1097	BITSTRING	1		RESERVED FOR FUTURE USE
(44A)	1098	SIGNED	2		RESERVED FOR FUTURE USE
(44C)	1100	SIGNED	4	TDNSSLHK	HANDSHAKE TYPE
		....		TDNSSLHC	"0" HANDSHAKE TYPE: CLIENT
		1		TDNSSLHS	"1" HANDSHAKE TYPE: SERVER
		.... ..1.		TDNSSLHA	"2" HANDSHAKE TYPE: CLIENT AUTH
		11		TDNSSLHN	"3" HANDSHAKE TYPE: NO CLI AUTH
(450)	1104	SIGNED	4	TDNSSLCP	CIPHER LEVEL
		1		TDNSSLCL	"1" CIPHER LEVEL: WEAK
		.... ..1.		TDNSSLCH	"2" CIPHER LEVEL: STRONG
(454)	1108	ADDRESS	4		RESERVED FOR FUTURE USE
(458)	1112	ADDRESS	4		RESERVED FOR FUTURE USE
(45C)	1116	ADDRESS	4		RESERVED FOR FUTURE USE
(460)	1120	ADDRESS	4		RESERVED FOR FUTURE USE
(464)	1124	ADDRESS	4		RESERVED FOR FUTURE USE
(468)	1128	ADDRESS	4		RESERVED FOR FUTURE USE
(46C)	1132	ADDRESS	4		RESERVED FOR FUTURE USE
		..11		TDNSSLDL	"*-TDNSSLDS" LENGTH OF WORKAREA
	AREAS USED FOR SOCKET CALL : SELECTEX LIST OF SOCKET DESCRIPTORS TO BE PROCESSED				

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(470)	1136	ADDRESS	4	TDCBSXRI (21)	READLIST: INPUT
(4C4)	1220	ADDRESS	4	TDCBSXRO (21)	READLIST: OUTPUT
(518)	1304	ADDRESS	4	TDCBSXWI (21)	WRITELIST: INPUT
(56C)	1388	ADDRESS	4	TDCBSXWO (21)	WRITELIST: OUTPUT
(5C0)	1472	ADDRESS	4	TDCBSXXI (21)	EXCEPTION: INPUT
(614)	1556	ADDRESS	4	TDCBSXXO (21)	EXCEPTION: OUTPUT
(614)	1556		0	TDCBSXAL	"(*-TDCBSXRI)" LENGTH OF BIT ARRAYS
(614)	1556		0	TDCBLN	"(*-TDCBSD)" LENGTH OF TDCB/SDCB
EQUATES FOR CALLING MODULES IPW$$TS, IPW$$SS					
		....		INTFTSMS	"0" CALL SERVICE FOR MESSAGE PROCESSING
		.... .1..		INTFTSSO	"4" CALL SERVICE FOR SOCKET CALLS
		.... 1...		INTFTSTX	"8" CALL SERVICE FOR TIMER INTERVAL - STXIT= HANDLING
		.... 11..		INTFTSTT	"12" CALL SERVICE FOR TIMER INTERVAL - TIME= HANDLING
		...1		INTFTSTC	"16" CALL SERVICE FOR TIMER INTERVAL - CANCEL=HANDLING
		...1 .1..		INTFTSTP	"20" CALL SERVICE FOR TIMER INTERVAL - PROCESS=HANDLING
		...1 1...		INTFTSRC	"24" CALL SERVICE FOR SOCKET RC CHECKING

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
					TCP/IP CONTROL RECORD
(0)	0	STRUCTURE	0	TCPCTRL	, DUMMY SECTION DEFINITION
(0)	0	CHARACTER	8	TCPCTTY	TYPE OF CONTROL RECORD
(8)	8	CHARACTER	8	TCPCTRH	FROM NJE NODENAME
(10)	16	BITSTRING	4	TCPCTRI	FROM IP ADDRESS
(14)	20	CHARACTER	8	TCPCTOH	TO NJE NODENAME
(1C)	28	BITSTRING	4	TCPCTOI	TO IP ADDRESS
(20)	32	BITSTRING	1	TCPCTRC	REASON CODE
		..1. ...1		TCPCTRLN	"*-TCPCTRL" LENGTH OF CONTROL RECORD

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
					TCP/IP BLOCK HEADER
(0)	0	STRUCTURE	0	TCPTTB	, DUMMY SECTION DEFINITION
(0)	0	BITSTRING	1	TCPTTBF	FLAG BYTE
(1)	1	BITSTRING	1	TCPTTBU	UNUSED
(2)	2	BITSTRING	2	TCPTTBLN	LENGTH (INCL. TTB, TTR, TTREOB)
(4)	4	BITSTRING	4		UNUSED
		.... 1...		TCPTTBLL	"*-TCPTTB" LENGTH OF TTB

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
					TCP/IP BLOCK RECORD HEADER
(0)	0	STRUCTURE	0	TCPTTR	, DUMMY SECTION DEFINITION
(0)	0	BITSTRING	1	TCPTTRF	FLAG BYTE
(1)	1	BITSTRING	1	TCPTTRU	UNUSED
(2)	2	BITSTRING	2	TCPTTRLN	LENGTH (TTR NOT INCLUDED)
		.... .1..		TCPTTRLL	"*-TCPTTR" LENGTH OF TTR

Print Status Processor Work Area

Definition Macro: IPW$DEF PSWRKA=YES

This work area is used to pass information from the command processor to the print status processor, and to control the processing of the print status task.

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
				PRINT STATUS PROCESSOR WORKAREA	
(0)	0	CHARACTER	32	PSWSD (0)	STORAGE DESCRIPTOR
(0)	0	CHARACTER	28		
(1C)	28	ADDRESS	4	PSWTPTR	POINTER TO TCB OWNING PS WORKAREA
	COMMUNICATION AREA				
(20)	32	CHARACTER	2	PSWFLGS (0)	VARIOUS SWITCHES
(20)	32	BITSTRING	1	PSWFLG1	FLAG BYTE 1
		1...		PSWSUPR	"X'80'" .. TURNED ON BY CALLER OF 'BINTODEC' SUBROUTINE TO INDICATE THAT SUPPRESSION OF LEADING ZEROS IS RE- QUESTED.
		.1..		PSWFND	"X'40'" .. TURNED ON WHEN A QUEUE RECORD IS FOUND WHICH IS ELIGIBLE TO BE DISPLAYED
		..1.		PSWPRNT	"X'20'" .. TURNED ON WHEN THE OUTPUT IS DESTINED FOR A PRINTER DEVICE
		...1		PSWSPOOL	"X'10'" .. TURNED ON IF THE OUPUT IS SUPPOSED TO BE SPOOLED TO DISK AS LST QUEUE ENTRY.
		.... 1...		PSWCTRL	"X'08'" .. TURNED ON WHEN THE OP CODE IN THE CCW REPRESENTS AN IMMEDIATE CMND
		.... .1..		PSWRMTT	"X'04'" .. TURNED ON WHEN THE ISSUER OF THE PDISPLAY COMMAND IS A REMOTE OP HOOKED UP TP A REMOTE SYSTEM.
		.... ..1.		PSWFMSG	"X'02'" .. TURNED ON WHEN THE QUEUE DISPLAY SHOULD RETURN FIXED FORMAT MSG"S
		1		PSWONE	"X'01'" .. TURNED ON WHEN AT LEAST ONE QUEUE ENTRY IS DISPLAYED
(21)	33	BITSTRING	1	PSWFLG2	FLAG BYTE 2
		1...		PSW2NCC	"X'80'" .. DO NOT GENERATE CNTL CMD
		.1..		PSWPEF	"X'40'" .. PNET ENTRY FOUND INDICATION
		..1.		PSWCLR	"X'20'" .. CTLSPOOL LOOK UP REQUEST
		...1		PSWF2FT	"X'10'" .. FIRST TIME SWITCH TURNED ON AFTER THE 1ST TAPE ENTRY IS DISPLAYED
		.... .1..		PSWDUE5	"X'04'" .. 5TH FULL=YES LINE TO DO
		.... ..1.		PSWDUEC	"X'02'" .. DUE LIST IN CONTIN. MODE
		1		PSWCS	"X'01'" .. CHANNEL CMD ALREADY SET
	PDISPLAY ARGUMENT LIST				
			THE FOLLOWING FIELDS REPRESENT THE ARGUMENT LIST WHICH WILL BE PASSED TO THE PRINT STATUS TASK TO PERFORM THE APPROPRIATE DISPLAY FUNCTIONS. THE FIRST 28 BYTES ARE USED AS COMMOM PART BY PDISPLAY XX AS WELL AS BY PDISPLAY PNET OR DYNC.		
(24)	36	SIGNED	4	PSWDARGL (0)	ARGUMENT LIST
(24)	36	CHARACTER	1	PSWDID	PARAMETER LIST FLAG BYTE
		CHARACTER		PSWDDID	"C'D'" .. ID FOR DEFAULT DISPLAY
		CHARACTER		PSWDPID	"C'P'" .. ID FOR PNET DISPLAY

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
		CHARACTER		PSWDVID	"C'Q'" .. ID FOR CORE COPY DISPLAY
		CHARACTER		PSWDTID	"C'T'" .. ID FOR TAPE DISPLAY
		CHARACTER		PSWDYID	"C'Y'" .. ID FOR DYNC DISPLAY
		CHARACTER		PSWDSID	"C'S'" .. ID FOR STATISTICS DISPLAY
		CHARACTER		PSWDEID	"C'E'" .. ID FOR EXIT DISPLAY
(25)	37	BITSTRING	1	PSWDTOID	DESTINATION OF REPORT
(26)	38	BITSTRING	2	PSWDLU	LOGICAL UNIT # OF PRINTER
(28)	40	ADDRESS	4	PSWDECB	ECB ADDRESS
(2C)	44	ADDRESS	4	PSWDTCB	ADDR OF REQUESTING TASK TCB
(30)	48	BITSTRING	2		RESERVED
(32)	50	BITSTRING	1	PSWDFLG1	FLAG1 BITS:
		1...		PSWD1DUE	"X'80'" - PDISPLAY CDUE=
		.1..		PSWD1FRR	"X'40'" - PDISPLAY RDR,FREER
		..1.		PSWDDEX	"X'20'" - PDISPLAY EXIT DATA FND
(33)	51	BITSTRING	1	PSWDFLG	FLAG BITS:
		1...		PSWDREM	"X'80'" - PDISPLAY RJE
		.1..		PSWDHLD	"X'40'" - PDISPLAY HOLD
		..1.		PSWDFRE	"X'20'" - PDISPLAY FREE
		...1		PSWDLOC	"X'10'" - PDISPLAY LOCAL
		.... 1...		PSWDCON	"X'08'" - DISPLAY TARGET = CON
		.... .1..		PSWDPRT	"X'04'" - DISPLAY TARGET = PRT
		.... ..1.		PSWDLST	"X'02'" - DISPLAY TARGET = SPOOL LST QUEUE ENTRY
		1		PSWDFUL	"X'01'" - PDISPLAY ..FULL=YES
(34)	52	CHARACTER	9	PSWDFNM (0)	FROM NODE NAME + SYSID
(34)	52	CHARACTER	8	PSWDFNMN	.. FROM NODE NAME
(3C)	60	CHARACTER	1	PSWDFNMS	.. FROM SYSID
(3D)	61	CHARACTER	8	PSWDUID	FROM USER/REMOTE ID
(45)	69	BITSTRING	1	PSWDNMRF	FLAG BYTE FROM NMR
(46)	70	BITSTRING	1	PSWDFG2	COPY OF CPFG2 FLAG BYTE
(47)	71	BITSTRING	1		RESERVED
(48)	72	SIGNED	4	PSWDPPA	COPY OF $ICP PASS VALUE
(4C)	76	ADDRESS	4	PSWDBS	BEGIN SCAN INDICATOR
(50)	80	BITSTRING	1	PSWDQID	QUEUE PROCESSING FLAGS X'80' .. RDR QUEUE DISPLAY X'40' .. LST QUEUE DISPLAY X'20' .. PUN QUEUE DISPLAY X'10' .. XMT QUEUE DISPLAY
		.... 1...		PSWDQIDW	"X'08'" .. WFR SUBQUEUE DISPLAY
(51)	81	BITSTRING	2	PSWDJN	JOBNUMBER
(53)	83	BITSTRING	1	PSWDBIN	REMOTE ID (BINARY FORMAT)
(54)	84	BITSTRING	1	PSWDGJL	LENGTH OF GENERIC JOBNAME
(55)	85	CHARACTER	8	PSWDGJN	GENERIC JOBNAME
(5D)	93	CHARACTER	8	PSWDJOB	JOBNAME
(65)	101	BITSTRING	8	PSWDTNN	TARGET NODE NAME
(6D)	109	BITSTRING	1	PSWDCDP	CURRENT DISPOSITION
(6E)	110	BITSTRING	1	PSWDCPY	CURRENT PRIORITY
(6F)	111	BITSTRING	1	PSWDCSY	CURRENT SYSTEM ID
(70)	112	CHARACTER	4	PSWDCFI	CURRENT FORMS ID (FNO)
(74)	116	CHARACTER	8	PSWDTUS	CURRENT 'TO' USER ID
(7C)	124	CHARACTER	8	PSWDFNN	'FROM' NODE NAME

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(84)	132	CHARACTER	8	PSWDFUS	'FROM' USER ID
(8C)	140	CHARACTER	1	PSWDCLS	JOBCLASS
(8D)	141	BITSTRING	1	PSWDCIX	PDISPLAY CLASS INDEX
(8E)	142	BITSTRING	2		RESERVED
(90)	144	CHARACTER	8	PSWDWRK	WORK FIELD
(98)	152	CHARACTER	8	PSWDDTE	CURRENT DATE 'CCYYMMDD'
(A0)	160	BITSTRING	1	PSWDMSK	BRANCH MASK
(A1)	161	CHARACTER	8	PSWDUSER	'FROM' OR 'TO' USER ID
(A9)	169	BITSTRING	1	PSWDCQID	CURRENT PROCESSED QUEUE $$PS
(AA)	170	BITSTRING	2		RESERVED
(AC)	172	ADDRESS	4	PSWDTPUB	TAPE UNIT PUB ENTRY ADDR
(B0)	176	SIGNED	2	PSWDTPUU	TAPE UNIT PROG.LOG.NUMBER
(B2)	178	CHARACTER	3	PSWDTCUU	TAPE UNIT CUU (EBCDIC)
(B5)	181	BITSTRING	3		RESERVED
(B8)	184	ADDRESS	4	PSWDPPUB	PRT UNIT PUB ENTRY ADDR
(BC)	188	SIGNED	2	PSWDPPUU	PRT UNIT PROG.LOG.NUMBER
(BE)	190	CHARACTER	3	PSWDPCUU	PRT UNIT CUU (EBCDIC)
(C1)	193	BITSTRING	3		RESERVED
(C4)	196	ADDRESS	4	PSWSDSPY	SAVE ADDRESS DST BLOCKS
(C8)	200	CHARACTER	16	PSWDCUIN	CURRENT USER INFO
		1.11 .1..		PSWDARLN	"*-PSWDARGL" ARGUMENT LIST LENGTH

REGISTER SAVE AREA USED BY IPW$$PS1

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(D8)	216	SIGNED	4	PSWREGS1 (15)	IPW$$PS1 REG SAV AREA R0-RE
(114)	276		0	PSWKHDLN	"*-PSWADS" LENGTH OF COM. AREA+ARG.LIST

VARIABLES USED BY PDISPLAY PNET COMMAND

THE FOLLOWING FIELDS TOGETHER WITH THE COMMON PART OF THE ARGUMENT LIST REPRESENT THE ARGUMENT LIST AS IT IS PASSED TO THE PRINT STATUS TASK TO PERFORM THE PNET DISPLAY FUNCTIONS.
NOTE: THE SECTION MUST BE MAINTAINED TOGETHER WITH THE APPROPRIATE SECTION IN THE CP WORKAREA.

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(4C)	76	SIGNED	4	PSWDPPTR	POINTER TO THE SPECIFIED NODE ID
(50)	80	BITSTRING	1	PSWPFLG1	FLAG BYTE1
		1...		PSWDPOWN	"X'80'" .. OWN NODE DISPLAY REQUEST
		.1..		PSWDPLIN	"X'40'" .. LINK DISPLAY REQUEST
		..1.		PSWDPNID	"X'20'" .. SPECIFIES NODE DISPLAY REQUEST
		...1		PSWDPALL	"X'10'" .. ALL NODES DISPLAY REQUEST
(51)	81	BITSTRING	1		UNUSED
(52)	82	CHARACTER	8	PSWNODID	NAME OF NODEID

VARIABLES USED BY PDISPLAY DYNC/STATUS COMMAND

THE FOLLOWING FIELDS TOGETHER WITH THE COMMON PART OF THE ARGUMENT LIST REPRESENT THE ARGUMENT LIST AS IT IS PASSED TO THE PRINT STATUS TASK TO PERFORM THE DYNC DISPLAY OR THE STATISTICS DISPLAY FUNCTIONS.
NOTE: THE SECTION MUST BE MAINTAINED TOGETHER WITH THE APPROPRIATE SECTION IN THE CP WORKAREA.

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(4C)	76	SIGNED	4	PSWDDPPA	POINTER TO DCLT AREA
(50)	80	SIGNED	4	PSWDDNUM	JOB NO. OF LIST QUEUE ENTRY
(54)	84	SIGNED	4	PSWDDLMG	NO. OF TERMINATING MESSAGE
(58)	88	BITSTRING	1	PSWDDFL1	FLAG BYTE WITH CPFG SETTING
(59)	89	BITSTRING	1	PSWDDFL2	FLAG BYTE 2

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
		1...		PSWD2ALL	"X'80'" DISPLAY ALL
		.1..		PSWD2ENA	"X'40'" DISPLAY ALL ENABLED
		..1.		PSWD2DIS	"X'20'" DISPLAY ALL DISABLED
		...1		PSWD2INV	"X'10'" DISPLAY ALL INVALID
		.... 1...		PSWD2CLS	"X'08'" DISPLAY SINGLE CLASS
		.... .1..		PSWD2ACT	"X'04'" DISPLAY ACTIVE DCLT
		.... ..1.		PSWD2ONE	"X'02'" ONE LINE DISPLAYED
		1		PSWD2HED	"X'01'" HEAD LINE ALREADY DISPLAYED
(5A)	90	BITSTRING	1	PSWDDFL3	FLAG BYTE 3
(5B)	91	BITSTRING	1	PSWDDCLS	CLASS TO BE DISPLAYED
(5C)	92	SIGNED	4	PSWDLIMB	MSG LIMIT POSITION
(60)	96	BITSTRING	1	PSWDTYPI	DEVICE TYPE INDICATOR
REGISTER SAVE AREA USED BY SUBROUTINES					
(114)	276	SIGNED	4	PSWREG (12)	REG SAVE AREA RE-R5 FOR SUBR.
(144)	324	SIGNED	4	PSWREGMG (14)	REG SAVE AREA RE-R7 FOR PSMSG SUBROUTINE
(17C)	380	SIGNED	4	PSWREGQL (14)	REG SAVE AREA RE-R7 FOR PSQLU00 SUBROUTINE
(1B4)	436	SIGNED	4	PSWSRE	SAVE AREA FOR REGISTER 14
(1B8)	440	SIGNED	4	PSWSRTN	SAVE AREA FOR REGISTER 14 (MSG RTN)
(1BC)	444	SIGNED	4	PSWDUEF	SAVE AREA RETURN REG. 15
(1C0)	448	SIGNED	4	PSWDUE9	SAVE AREA $$PS 1ST BASE REG.
INTERFACE AREA FOR LOGICAL OUTPUT					
(1C4)	452	SIGNED	3	PSWLOPL (0)	LOGICAL OUTPUT PARAMETER LST
(1C4)	452	SIGNED	4	PSWLOJHR	..JOB HEADER RECORD
(1C8)	456	SIGNED	4	PSWLODHR	..DATA SET HEADER REC
(1CC)	460	SIGNED	4	PSWLOJTR	..JOB TRAILER REC (=ZERO)
CCB AND CCW					
(1D0)	464	DBL WORD	8	(0)	FORCE DOUBLEWORD ALIGNMENT
(1D0)	464	BITSTRING	16	PSWCCB	PRINT CCB
(1E0)	480	BITSTRING	8	PSWCCW	PRINT CCW
VARIABLES USED FOR SUBROUTINE 'BINTODEC'					
(1E8)	488	DBL WORD	8	PSWDBLW	DOUBLE WORD USED FOR CONVERSION
(1F0)	496	CHARACTER	10	PSWVDEC	CONTAINS DECIMAL NUMBER IN PRINTABLE FORMAT
(1FA)	506	CHARACTER	2		RESERVED FOR FUTURE USE
VARIABLES USED FOR PRINTING DUE TIME INFORMATION					
(1FC)	508	CHARACTER	3	PSFPACK (0)	AREA TO BE UNPACKED
(1FC)	508	CHARACTER	1	PSFPK1	DATA BYTE 1 TO BE UNPACKED
(1FD)	509	CHARACTER	1	PSFPK2	DATA BYTE 2 TO BE UNPACKED
(1FE)	510	CHARACTER	1	PSFPK3	SIGN BYTE TO BE UNPACKED
(1FF)	511	BITSTRING	1		RESERVED FOR FUTURE USE
(200)	512	SIGNED	4	PSWDMAX	TO SAVE LIST-LENGTH PLUS 1
NODAL MESSAGE RECORD (NMR)					
(204)	516	CHARACTER	2		USED FOR ALIGNMENT
(206)	518	BITSTRING	1	PSWNMR (0)	NODAL MESSAGE RECORD
(206)	518	BITSTRING	30		SYSTEM PREFIX
(224)	548	CHARACTER	88	PSWPMSG (0)	MESSAGE OUTPUT/MODIFICATION AREA
(224)	548	BITSTRING	1	PSWMLN	LENGTH OF MESSAGE
(225)	549	CHARACTER	87	PSWMSG (0)	MESSAGE AREA

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(225)	549	CHARACTER	7		MESSAGE IDENT. (BE EVEN 6)
(22C)	556	CHARACTER	80	PSWTXT	MESSAGE TEXT
(27C)	636	CHARACTER	80	(PSWTXTJ)	- (MESSAGE TEXT FOR JOURNAL) - TO BUILD FIX FORMAT MSG EX1
(2CC)	716	CHARACTER	80		TO BUILD FIX FORMAT MSG EX2
TAPE DISPLAY WORK AREA					
(31C)	796	BITSTRING	1	PSWTQID	IND OF QUEUES-TO-PROCESS ON TAPE (SAME FLAGS AS FOR PSWDQID)
(31D)	797	BITSTRING	1	PSWTQNFN	IND OF QUEUE FOR WHICH 'NOT FOUND' MESSAGE MUST BE ISSUED
(31E)	798	BITSTRING	1	PSWTQFN	INDICATOR FOR 'QUEUES FOUND' OR FOR 'MSG NOT-FOUND ALREADY ISSUED' (SAME FLAGS AS FOR PSWDQID)
(31F)	799	BITSTRING	1	PSWTWK	WORK FIELD
POFFLOAD JOURNAL WORK AREA					
(31E)	798	BITSTRING	1	PSWJQFN	INDICATOR FOR 'QUEUES FOUND' (SAME FLAGS AS FOR PSWDQID)
(31F)	799	BITSTRING	1	PSWJWK	WORK FIELD
MISCELLANEOUS					
(320)	800	ADDRESS	4	PSWCPTR	POINTS TO BEGIN OF CLASS TABLE
(324)	804	SIGNED	2	PSWCLLC	NUMBER OF SCANS TO BE PERFORMED
(326)	806	SIGNED	2	PSWLCNT	LINE COUNT
(328)	808	BITSTRING	1	PSWSLDT	UNIT TYPE OF SYSLOG DEVICE
(329)	809	BITSTRING	3		UNUSED
(32C)	812	ADDRESS	4	PSWRDCL	POINTS TO Q-REC COLLECTION
(330)	816	BITSTRING	8		UNUSED
VARIABLES USED BY QUEUE RECORD DUMP ROUTINE					
(338)	824	CHARACTER	12	PSWIOR (0)	IO-REQUEST WORD
(338)	824	SIGNED	4	PSWQCW	REL. QUEUE REC. NUMBER
(33C)	828	SIGNED	4	PSWAQR	QUEUE RECORD AREA ADDRESS
(340)	832	SIGNED	2		AREA LENGTH
(342)	834	CHARACTER	1		READ/WRITE CODE QUEUE FILE
(343)	835	BITSTRING	1		FLAG BYTE
(344)	836	SIGNED	2	PSWQRB	REL.QUEUEREC.-BLOCKNUMBER
(348)	840	ADDRESS	4	PSWAIND	POINTER TO ADDR. OF SLICE
(34C)	844	SIGNED	2	PSWQMB	NUMBER OF RECORDS PER FORM
(34E)	846	SIGNED	2	PSWQPR	NUMBER OF PRINTED RECORDS
(350)	848	SIGNED	2	PSWREM	NUMBER OF BYTES EDITED AND PRINTED * AT A TIME
(352)	850	SIGNED	2	PSWPG	PAGE NUMBER
(354)	852	CHARACTER	5	PSWUPF	UNPACK FIELD
(35A)	858	SIGNED	2	PSWH	USED TO CALC. DISPLACEMENT
ALIGN TO LINE BOUNDARY AND FILL UP WITH ZEROS					

Printer TCB Extension Area

Definition Macro: IPW$DTE

This control block is constructed:

- At job execution time whenever a printer device is being spooled.
- At print time.

Like the TCB, the printer TCB extension area exists as long as the task exists. The control block contains device status information of the current or new printer setup. The TCB extension is pointed to by the TC3E field in the TCB.

Bytes Hex.	Label of Field	Description/Function
00-07	PTEFCBN	FCB phase name, loaded on the printer
08-0F	PTEUCSN	UCS buffer phase name
10	PTEUCSO	UCS option byte
	PTEUCSOC	X'80' - UCS data check option
	PTEUCSOF	X'40' - UCS fold option
11	PTEFLAG	Status byte
	PTECLRPR	X'80' - Clear printer at end of job
	PTE3800x	X'01' - 3200 printer
	PTE4248	X'40' - Horizontal copy requested
	PTEBAND	X'20' - Band id check requested
12-15		Reserved
16-17	PTEDEV	Logical unit number or device address
18-1A		Reserved

- 3800 Printer Control Information

1B	PTE3CTRC	Current TRC command
1C	PTE3RQB	Pending request byte
	PTE3SRI	X'80' - SETPRT required indicator
	PTE3TRC	X'01' - OPTCD=J specified
1D	PTE3CGI	Current copy group index
1E	PTE3CSTT	Current translate table op. code
1F	PTE3NTRC	New TRC indicator

- General Work Area

20-3F	PTEGWA	General work area

- SETPRT Parameter List

40-83	PTELIST	SETPRT parameter list as generated by the SPLIST macro.
	PTELN	Length of printer extension area

Queue Record Area (QRA)

Definition Macro: IPW$DQR

This area is used in conjunction with the auxiliary queue record area in the disk management block. Each task that processes a queue record acquires a QRA to contain the record.

The format as it is printed in a dump is as follows.

Refer to the Disk Management Block (DMB) auxiliary queue record area for a fuller description of the individual entries.

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
QUEUE RECORD AREA (QRA)					
THE BODY OF THE QUEUE RECORD					
					CONTAINS INFORMATION PERTINENT TO THIS PARTICULAR QUEUE ENTRY AND THE USER JOB WHICH CREATED IT. >>> NOTE: THE PLS CODE IS DUPLICATED IN IPW$DQC. BE SURE TO MAINTAIN BOTH COPIES.
(0)	0	CHARACTER	256	QRPT1 (0)	QUEUE RECORD PART 1
(0)	0	CHARACTER	136	QRBF (0)	BODY FIELDS
(0)	0	CHARACTER	8	QRDY	DATE (CREATING SYST. FORMAT)
(8)	8	CHARACTER	35	QRSA (0)	INTERNAL REFERENCE FIELD
(8)	8	CHARACTER	4	QRST	OPERATION START TIME
(C)	12	CHARACTER	4	QRET	OPERATION END TIME
(10)	16	CHARACTER	16	QRUI	USER INFORMATION
(20)	32	CHARACTER	8	QRNM	JOB NAME
(28)	40	SIGNED	2	QRJNO	JOB NUMBER
(2A)	42	BITSTRING	1	QRQI	QUEUE RECORD IDENTIFIER
		CHARACTER		QRIR	"C'R'" ..RDR IDENTIFIER
		CHARACTER		QRIL	"C'L' ..LST IDENTIFIER @D35BIQI
		CHARACTER		QRIP	"C'P'" ..PUN IDENTIFIER
		CHARACTER		QRIC	"C'C'" ..CONSOLE IDENT(WRITER-ONLY)
		CHARACTER		QRIF	"C'F'" ..QUEUE REC IS MARKED FREE
		CHARACTER		QRID	"C'D'" ..QUEUE REC IS MARKED LAST
		CHARACTER		QRII	"C'I'" ..QUEUE REC IS MARKED INTERN
		CHARACTER		QRIB	"C'B'" ..QUEUE REC IS MARKED BAD
(2B)	43	BITSTRING	1	QRCN	VSE/POWER CANCEL CODE
		...1		QRCNM	"X'10'" .. NORMAL END OF JOB
		..1.		QRCCC	"X'20'" .. PCANCEL WAS ISSUED
		..11		QRCSP	"X'30'" .. PSTOP COMMAND ISSUED
		.1..		QRCFL	"X'40'" .. PFLUSH COMMAND ISSUED
		.1.1		QRCDL	"X'50'" .. PDELETE OR PURGE ISSUED
		.11.		QRCRX	"X'60'" .. FLUSHED VIA READER EXIT
		.111		QRCIO	"X'70'" .. CANCELED DUE TO I/O ERROR

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
		1...		QRCRC	"X'80'" .. PNET RECEIVER CANCEL
		1..1		QRCQT	"X'90'" .. QUIT REQUEST FROM X-PART
		1.1.		QRCSE	"X'A0'" .. CANCELED DUE TO SEV ERROR
		1.11		QRCCL	"X'B0'" .. GET_CLOSE REQ FROM X-PART
		11..		QRCSR	"X'C0'" .. CANCELED DUE TO SOD
		11.1		QRCPF	"X'D0'" .. PRINTING/PUNCHING FAILED
(2C)	44	BITSTRING	1	QRRJ	LINE IDENTIFIER
		..1. 11..		QRDT	"QRRJ" DEVICE TYPE
(2D)	45	CHARACTER	3	QRCU	CHANNEL AND UNIT (LINE ADDRESS)
(30)	48	BITSTRING	1	QRFJ	FROM TERMINAL IDENTIFIER
(31)	49	BITSTRING	1	QRTJ	TO TERMINAL IDENTIFIER
(32)	50	CHARACTER	1	QRCL	CLASS
(33)	51	CHARACTER	1	QRPY	PRIORITY
(34)	52	SIGNED	4	QRNR	RECORD COUNT
(38)	56	SIGNED	1	QRPYSL	PRIORITY - SAVED LOCAL
(39)	57	SIGNED	1	QRUEX	USER EXIT WORK BYTE - MUST NOT BE USED BY VSE/POWER
(3A)	58	BITSTRING	1	QRSN	JOB SUFFIX NUMBER
		1...		QRSNLA	"X'80'" .. LAST SEGMENT INDICATOR
(3B)	59	SIGNED	1	QRNC	NUMBER OF COPIES
(3C)	60	CHARACTER	4	QRFI	FORMS IDENTIFIER
(40)	64	SIGNED	4	QRCREC	CHECKPOINT RECORD NUMBER
(44)	68	CHARACTER	2	QRDYC	CENTURY OF CREATION DATE
(46)	70	ADDRESS	1	QRCCPY	CHECKPOINT COPY NUMBER
(47)	71	BITSTRING	1	QRDGP0	DUE DATE GENERAL BYTE 0
		1...		QRDG0X	"X'80'" DUE DATE INFO EXISTS
		.1..		QRDG0W	"X'40'" ENTRY QUEUED IN WFR-SQ
(48)	72	SIGNED	4	QRLC	LINE/CARD COUNTER
(4C)	76	SIGNED	4	QRRR	RESTART PAGE COUNT
(50)	80	SIGNED	1	QRCR	COPIES REMAINING
(51)	81	CHARACTER	1	QRDI	NEW DISP OR PURGE/FLUSH IND
		CHARACTER		QRDIP	"C'P'" ..PURGE/FLUSH QREC
		CHARACTER		QRDIH	"C'H'" ..'HOLD' QUEUE SET FLAG
(52)	82	CHARACTER	1	QRDP	DISPOSITION
		CHARACTER		QRDPD	"C'D'" .. DISPATCHABLE DISPOSITION
		CHARACTER		QRDPK	"C'K'" .. 'KEEP' DISPOSITION
		CHARACTER		QRDPL	"C'L' .. 'LEAVE' DISPOSITION
		CHARACTER		QRDPH	"C'H'" .. 'HOLD' DISPOSITION
		CHARACTER		QRDPN	"C'N'" .. 'NO SPOOLING' DISP C'T' .. SPOOL TAPE
(53)	83	SIGNED	1	QRSP	NUMBER OF SEPARATORS
(54)	84	SIGNED	4	QRBS	NUMBER OF RECORDS BEFORE SPLIT
(58)	88	SIGNED	4	QRBM	MAXIMUM VALUE OF COUNT
(5C)	92	SIGNED	4	QRBN	ADDITIONAL COUNT VALUE
(60)	96	BITSTRING	2	QRER	3540 UNIT SPECIFICATION FOR OUTPUT QUEUE ENTRIES FROM XW, THE ABOVE FIELD IS USED TO SAVE THE PAGE LENGTH.
(62)	98	SIGNED	2	QRJ#	SAVE JOB NUMBER FOR ACCNT
(64)	100	CHARACTER	4	QRCP	COMPACTION TABLE NAME

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
3800 PRINTER CONTROL INFORMATION OVERLAYED BY DUE DATE INFO (ONLY FOR RDR POSSIBLE)					
(68)	104	CHAR-ACTER	4	QRFL	FORMS OVERLAY IDENTIFIER
(6C)	108	BITSTRING	8	QRCG	COPY GROUPS
(74)	116	BITSTRING	1	QRTC	TRANSMISSION COUNT
(75)	117	BITSTRING	1	QRCI	COPY GROUP INDEX
(76)	118	BITSTRING	1	QRPS	PAPER STATUS
		CHAR-ACTER		QRBR	"C'B'" .. BURST REQUEST
CONTINUATION OF GENERAL SECTION					
(77)	119	BITSTRING	1	QROP	GENERAL OPTION BYTE 1
					(NOTE - MOST BITS ARE DEFINED IN THE DMB FIELD MROP)
					X'80' ..(CLEAR PRINT AT EOJ)
					X'40' ..(MARK FORM FOR SEP PAGES)
		..1.		QRCS	"X'20'" .. NO SEP PAGES BTWN COPY
		...1		QROHP	"X'10'" .. HOLD WHEN PRT/PUN FAILS
					X'08' .. RESERVED
					X'04' .. RESERVED
					X'02' ..(CHANNEL 12 OPTION)
					X'01' ..(FEED OPTION 3540)
(78)	120	CHAR-ACTER	8	QRPW	PASSWORD
(80)	128	ADDRESS	2	QROJ#	ORIGINAL JOB NUMBER
(82)	130	CHAR-ACTER	1	QRSID	SYSID OF TARGET CPU
(83)	131	CHAR-ACTER	1	QRODP	ORIGINAL DISPOSITION
(84)	132	ADDRESS	2	QRRL	MAX RECORD LENGTH
(86)	134	BITSTRING	1	QRRCFM	RECORD FORMAT
		1...		QRRSCS	"X'80'" .. SCS PRINT
		.1..		QRRBMS	"X'40'" .. BMS MAPPING
		..1.		QR3270	"X'20'" .. 3270 FORMAT
		...1		QRRAPA	"X'10'" .. APA DATA FORMAT (CPDS)
		.... 1...		QRRESC	"X'08'" .. ESCAPE MODE
		.... .1..		QRRASA	"X'04'" .. ASA CARRIAGE CONTROL CHAR
		.... ..1.		QRRMCC	"X'02'" .. MACHINE CARRIAGE CONTROL
(87)	135	BITSTRING	1	QRVOL	Q-ENTRY LABELED TAPE FLAG
		1...		QRVLAST	"X'80'" .. LAST MULTI-VOLUME
		.111 1111			------- .. (VOLUME NUMBER)
					THE MAXIMUM VOLUME NUMBER IS 126.
					ANY VALUE OVER 126 MEANS GREATER OR EQUAL 127.
CONTROL SECTION					
THE CONTROL PORTION OF THE QUEUE RECORD CONTAINS INFORMATION RELATING TO THE STATUS OF THE QUEUE RECORD AND TO ITS POSITION WITHIN THE VSE/POWER QUEUES. NOTE: POFFLOAD LOAD/SELECT WILL COPY CERTAIN BYTES OF THIS SECTION. OTHER BYTES ARE NOT MAINTAINED.					
(88)	136	CHAR-ACTER	48	QRCF (0)	CONTROL FIELDS
(88)	136	CHAR-ACTER	1	QRXS	EXECUTION SWITCH
					C'X' ..ENTRY BEING PROCESSED
(89)	137	BITSTRING	1		RESERVED
(8A)	138	BITSTRING	1	QRRX	RESTART FUNCTION INDEX
(8B)	139	BITSTRING	1	QRSY	SYSTEM ID PROCESSING QR
(8C)	140	BITSTRING	1	QRS1	CONTROL FLAG BYTE 1
					NOTE: POFFLOAD LOAD/SELECT WILL COPY QRS1. BE CAREFUL
					THAT "EXECUTION-ONLY" FLAGS ARE TURNED OFF.
		1...		QRXQ	"X'80'" ..QUEUE SET IN XMIT QUEUE
		.1..		QR1AB	"X'40'" ..ABENDED ENTRY, DISP=X

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
		..1.		QR1AP	"X'20'" ..APPENDABLE ENTRY, DISP=A
		...1		QR1CK	"X'10'" ..CHECKPOINTED CRE-ENTRY
		.... 1...		QR1PF	"X'08'" ..PRT/PUN FAILED ENTRY, D=YD
		.... .1..		QR1NO	"X'04'" NT MUST NOT UPDATE NDHGLREC
		.... ..1.		QR1DD	"X'02'" ..DO NOT DELETE QUEUE ENTRY
		1		QR1NJN	"X'01'" ..ASSIGN NEW JOB NUMBER
(8D)	141	BITSTRING	1	QRS2	CONTROL FLAG BYTE 2 NOTE: POFFLOAD LOAD/SELECT WILL COPY QRS2. BE CAREFUL THAT "EXECUTION-ONLY" FLAGS ARE TURNED OFF.
		1...		QR2NP	"X'80'" ..JOB FROM NON-PNET NODE
		.1..		QR2EP	"X'40'" ..IN EXEC. PREPARATION PHASE
		..1.		QR2NNR	"X'20'" .. DO NOT UPDATE NDHGNREC
		...1		QR2UFR	"X'10'" .. ORIGIN USER BY 'FROM'
		.... 1...		QR2BTO	"X'08'" .. IGNORE BLANK TRUN.(VER.2)
		.... .1..		QR2XRD	"X'04'" ..BEING PROCESSED BY EX.RDR
		.... ..1.		QR2RUN	"X'02'" ..JOB IGNORE SET NORUN
(8E)	142	BITSTRING	1	QRS3	CONTROL FLAG BYTE 3
		1...		QR3PSH	"X'80'" ..POFFLOAD PICKUP SCHEDULED FOR ENTRY
		.1..		QR3DEL	"X'40'" ..QE IN DELAYED DELETE
		..1.		QR3NNC	"X'20'" ..I/O ERROR DURING NODE CHG
(8F)	143	BITSTRING	1	QRACN1	NON SHARED BROWSE COUNT OR SHARED SYS 1+2 BROWSE COUNT
(90)	144	ADDRESS	4	QRCRCT	PUT CHECKPOINT REC NUMBER
(94)	148	ADDRESS	4	QRRBC	CARDS/PAGES BEFORE CHKPT
(98)	152	BITSTRING	4	QRAC39	SHARED SYSID 3-9 BROWSE CNT.
(9C)	156	BITSTRING	8	QRADD	ADD 'STCK' STAMP
(A4)	164	ADDRESS	4	QRQP	PREVIOUS SET IN QUEUE
(A8)	168	ADDRESS	4	QRQN	NEXT SET IN QUEUE
(AC)	172	ADDRESS	4	QRDF	1ST DBLK NO OF 1ST DBLK GP
(B0)	176	ADDRESS	4	QRLDF	1ST DBLK NO OF LAST DBLK GP
(B4)	180	ADDRESS	4	QRNB	NO OF DBLK GROUPS USED
		1.11 1...		QR2L	
	EXTENSION -A OF THE BODY FIELDS X'B8' --> BODY FIELDS OF PART 2 --> X'FF'				
(B8)	184	CHARACTER	72	QRB2 (0)	BODY FIELDS EXTENSION -A
(B8)	184	CHARACTER	8	QRTN	TARGET NODE NAME
(C0)	192	CHARACTER	8	QRTU	TARGET USER ID
(C8)	200	CHARACTER	8	QRON	ORIGINATOR NODE NAME
(D0)	208	CHARACTER	8	QROU	ORIGINATOR USER NAME
		11.1 1...		QRV2L	"*-QRDS" RECORD LENGTH OF VERSION 2
(D8)	216	ADDRESS	4	QRWFRN	PTR TO NEXT WFR SUBQ ENTRY
	OVERLAY SECTION 1				
			(DIFFERENT USAGE FOR INPUT AND OUTPUT QUEUE ENTRIES) USED FOR OUTPUT QUEUE ENTRIES		
(DC)	220	CHARACTER	8	QROUT1 (0)	OUTPUT RELATED FIELD
(DC)	220	CHARACTER	8	QRDIST	DISTRIBUTION CODE
			USED FOR INPUT QUEUE ENTRIES		
(DC)	220	CHARACTER	8	QRSECN	SECURITY NODEID
	CONTINUATION OF GENERAL SECTION				
(E4)	228	BITSTRING	1	QROP2	GENERAL OPTION BYTE 2
		1...		QRO2BT	"X'80'" IGNORE BLANK TRUNCATION
		.1..		QRO2MSG	"X'40'" ISSUE MESSAGE 1Q4DI

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
		..1.		QRO2LGNO	"X'20'" LOG=NO SPECIFIED
		...1		QRO2XXXX	"X'10'" UNUSED
		.... 1...		QRO2QCM	"X'08'" QUEUE COMPLETION MESSAGE
		.... .1..		QRO2MR	"X'04'" GCM R-MSG FOR PRELEASE
		.... ..1.		QRO2MQ	"X'02'" GCM R-MSG ACC. TO Q-RECORD
(E5)	229	BITSTRING	1	QRFLGO	FLAG BYTE FOR IN- & OUTPUT
		1...		QRCKI	"X'80'" .. CKP INFO EXISTS
		.1..		QRCKE	"X'40'" .. CKP INFO NOT AVAILABLE
		..1.		QRSAN	"X'20'" .. NOT SPOOL ACCESS PROTECTED
OVERLAY SECTION 2					
(DIFFERENT USAGE FOR INPUT AND OUTPUT QUEUE ENTRIES) USED FOR OUTPUT QUEUE ENTRIES					
(E6)	230	CHARACTER	18	QROUT (0)	OUTPUT RELATED FIELDS
(E6)	230	BITSTRING	1	QROTF1	OUTPUT FLAG BYTE 1
		1111			UNUSED
		.... 1...		QR1LM	"X'08'" LINE-MODE STATE
		.... .1..		QR1LMI	"X'04'" LINE-MODE-IDM/IMM STATE
		.... ..1.		QR1PM	"X'02'" PAGE-MODE STATE
		1		QR1PM8	"X'01'" PAGE-MODE STATE
(E7)	231	BITSTRING	1		RESERVED FOR FUTURE USE
(E8)	232	SIGNED	4	QRPGN	PAGE COUNT
(EC)	236	SIGNED	2		PRESERVE SPLDLREC PUT-APPEND
(EE)	238	SIGNED	2		RESERVED FOR FUTURE USE
(F0)	240	BITSTRING	8		RESERVED FOR FUTURE USE
USED FOR INPUT QUEUE ENTRIES					
(E6)	230	CHARACTER	18	QRINP (0)	INPUT RELATED FIELDS
(E6)	230	BITSTRING	1		RESERVED FOR FUTURE USE
(E7)	231	BITSTRING	17	QRMRIN (0)	GCM R-MSG FOR PRELEASE
(E7)	231	BITSTRING	1	QRMRSI	SYSID FOR GCM R-MSG
(E8)	232	CHARACTER	8	QRMRAP	APPL FOR GCM R-MSG
(F0)	240	CHARACTER	8	QRMRUS	USER FOR GCM R-MSG
		1111		QRV4L	"*-QRDS" RECORD LENGTH OF V4.X.,5.1.
CONTINUATION OF GENERAL SECTION					
(F8)	248	CHARACTER	1	QRTDP	TRANSMISSION DISPOSITION
(F9)	249	BITSTRING	7		RESERVED FOR FUTURE USE
QRECORD PART 2					
EXTENSION -A OF THE CONTROL SECTION X'100'--> CONTROL SECTION PART 2 --> x'11F' CLEARED BY IPW$RQS					
THE CONTROL PORTION OF THE QUEUE RECORD CONTAINS INFORMATION RELATING TO THE STATUS OF THE QUEUE RECORDS AND TO ITS POSITION WITHIN THE VSE/POWER QUEUES					
(100)	256	BITSTRING	32	QRC2 (0)	CONTROL FIELDS EXTENSION-A
RESTART TO ACTIVE RECORD CONTROL AREA					
(100)	256	ADDRESS	4	QROTC	ADDRESS OF OWNING TCB OF UPDATE OR CREATE TASK
(104)	260	BITSTRING	12	QRCC (0)	CURRENT RECORD COUNTS MANTAINED BY $$GD FOR UPDATE/BROWSE, BY $$PD FOR CR
(104)	260	BITSTRING	4	QRCCNR	INTERNAL RECORD COUNT
(108)	264	BITSTRING	4	QRCCLC	DATA RECORD COUNT
(10C)	268	BITSTRING	4	QRCCPG	PAGE COUNT (USED BY $$PD)
(110)	272	BITSTRING	16		RESERVED FOR FUTURE USE

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
EXTENSION -B OF THE CONTROL SECTION X'120'--> BODY FIELDS PART 3 --> x'16F'					
(120)	288	BITSTRING	80	QRB3 (0)	BODY FIELDS EXTENSION -B
(120)	288	BITSTRING	80		RESERVED FOR FUTURE USE
DUE DATE INFORMATION OVERLAYS: 3800 PRINTER CONTROL INFORMATION					
(68)	104	CHARACTER	15	QRDD (0)	START OF INFO
(68)	104	BITSTRING	1	QRDGP1	GENERAL PURPOSE BYTE 1
		1...		QRDG1R	"X'80'" RERUN=NO SPECIFIED
					X'40' RESERVED
					X'20' RESERVED
					X'10' RESERVED
		.... 1...		QRDG1T	"X'08'" DAILY SPECIFIED
		.... .1..		QRDG1W	"X'04'" WEEKDAYS SPECIFIED
		.... ..1.		QRDG1D	"X'02'" DAYS WITHIN MONTH
		1		QRDG1M	"X'01'" MONTHS SPECIFIED
		.... 111.		QRDG1C	"QRDG1T+QRDG1W+QRDG1D" CYCLING INFO ?
(69)	105	BITSTRING	1	QRDGP2	GENERAL PURPOSE BYTE 2
		1...		QRDG2F	"X'80'" 1ST TIME, NO NUMBER CHANGE
(6A)	106	CHARACTER	6	QRDCY (0)	START OF CYCLING INFO
(6A)	106	BITSTRING	2	QRDMY	MONTHS WITHIN YEAR LEFT ALIGNED: 80=JAN, 40=FEB, 20=MAR, ...
(6C)	108	BITSTRING	4	QRDDM	DAYS WITHIN MONTH LEFT ALIGNED: 80=1ST, 40=2ND, 20=3RD, ...
(70)	112	CHARACTER	6	QRDN (0)	START OF NEXT DUE DATE PACKED DECIMAL WITHOUT SIGN
(70)	112	CHARACTER	4	QRDNDT (0)	NEXT DUE DATE (W/O TIME)
(70)	112	BITSTRING	2	QRDNY	YEAR (1988-2087)
(72)	114	BITSTRING	1	QRDNM	MONTH (1-12)
(73)	115	BITSTRING	1	QRDND	DAY (1-31)
(74)	116	CHARACTER	2	QRDNT (0)	START OF NEXT DUE TIME
(74)	116	BITSTRING	1	QRDNTH	HOUR (0-23)
(75)	117	BITSTRING	1	QRDNTM	MINUTES (0-59)
(76)	118	BITSTRING	1		RESERVED
		 1111		QRDLEN	"*-QRDD" LENGTH OF DUE DATE
(6C)	108	BITSTRING	1	QRDDW	WEEKDAYS
		1...		QRDWMO	"X'80'" MONDAY
		.1..		QRDWTU	"X'40'" TUESDAY
		..1.		QRDWWE	"X'20'" WEDNESDAY
		...1		QRDWTH	"X'10'" THURSDAY
		.... 1...		QRDWFR	"X'08'" FRIDAY
		.... .1..		QRDWSA	"X'04'" SATURDAY
		.... ..1.		QRDWSU	"X'02'" SUNDAY
SOME DISPLACEMENTS FOR THE OLD VERSION OF THE QUEUE RECORD, I.E. VERSION 5.1 AND PREVIOUS ONES					
		.1.. .1..		QROVNP	"X'44'" PAGE NO. 2 BYTES ONLY

Note: The labels in a queue record vary according to the generated DSECT. The first two characters are queue record in a present queue record, QN in a "next" queue record, and QP in a "previous" queue record. In PL/S listing these characters are replaced by QREC.

How to Locate: Refer to Figure 152 on page 731 in Chapter 6, “Diagnostic Aids.”

Remote Message Control Block (MSCB)

Definition Macro: IPW$DMS

The remote message control block controls all access to the remote message queue. The block is created by the VSE/POWER initialization routine (IPW$$I7) if RJE processing (BSC and/or SNA) has been specified in the VSE/POWER generation macros.

Offset Hex	Type	Len	Name (Dim)	Description
(0)	CHARACTER	16	MSSD	SECTION DESCRIPTOR
(10)	DBL WORD	8	MSWW	WORK AREA
(18)	BITSTRING	1	MSFC	FREE CHAIN INDEX
(19)	BITSTRING	1	MSFI	FUNCTION INDICATOR
(1A)	BITSTRING	1	MSCI	CURRENT INDEX
(1B)	CHARACTER	1		RESERVED
(1C)	SIGNED	4	MSLW	LOCKWORD
(20)	CHARACTER	48	MSSV (0)	REGISTER SAVE AREA
(20)	SIGNED	4	MSRE	REGISTER 14
(24)	SIGNED	4	MSRF	REGISTER 15
(28)	SIGNED	4	MSR0	REGISTER 0
(2C)	SIGNED	4	MSR1	REGISTER 1
(30)	SIGNED	4	MSR2	REGISTER 2
(34)	SIGNED	4	MSR3	REGISTER 3
(38)	SIGNED	4	MSR4	REGISTER 4
(3C)	SIGNED	4	MSR5	REGISTER 5
(40)	SIGNED	4	MSR6	REGISTER 6
(44)	SIGNED	4	MSR7	REGISTER 7
(48)	SIGNED	4	MSR8	REGISTER 8
(4C)	SIGNED	4	MSR9	REGISTER 9
(50)	CHARACTER	8		RESERVED
(58)	SIGNED	4	MSMSRET	RETURN REGISTER FOR IPW$$MS
	.1.1 11..		MSLN	"*-MSDS" LENGTH OF CONTROL BLOCK

How to Locate: Refer to Figure 152 on page 731 in Chapter 6, “Diagnostic Aids.”

RJE Line Control Block (LCB)

Definition Macro: IPW$DLC

The line control block describes the line and its status. It contains an entire line account record, which is completed and written to the account file at SIGNOFF time. It also contains the terminal characteristics that are copied from the remote table in virtual storage at SIGNON time.

When the line is started by the central operator, an LCB is built for that line in real storage. It is not released before the line is stopped. One LCB always corresponds to each active line, independent of the number of reader and writer tasks operating on the line.

The format of an LCB as printed in a dump is defined below. The line control block also contains the CCB, a CCW string, and other information used to perform a line operation, such as mode bytes and sense information.

Offset Hex	Type	Len	Name (Dim)	Description
(0)	CHARACTER	8	LCBHEAD	LCB HEADER
	Line Account Record			
(8)	SIGNED	2	LCBACCT (0)	LINE ACCOUNT RECORD
(8)	CHARACTER	8	LCBDATE	SYSTEM DATE
(10)	CHARACTER	4	LCBSION	SIGNON TIME (0HHMMSSF) PACKED
(14)	CHARACTER	4	LCBSIOF	SIGNOFF TIME (0HHMMSSF) PACKED
(18)	CHARACTER	16	LCBUSER	USER INFORMATION
(28)	CHARACTER	8	LCBPSWD	LINE PASSWORD
(30)	SIGNED	2	LCBICNT	INVALID RESPONSES PER SESSION
(32)	CHARACTER	1	LCBRCID	LINE ACCOUNT RECORD IDENTIFIER
(33)	CHARACTER	1	LCBSCOD	SIGNOFF CODE
	1		SIGNOP	"X'01'" SIGNOFF BY REMOTE OPERATOR
	.... ..1.		SIGNCS	"X'02'" STOP LINE DUE TO CENTRAL STOP (PSTOP CUU)
	.... .1..		SIGNTO	"X'04'" SIGNOFF DUE TO TIMEOUT
	.... 1...		SIGNLE	"X'08'" SIGNOFF DUE TO LINE ERROR
	...1		SIGNEOJ	"X'10'" STOP LINE AT EOJ (PEND OR PSTOP ,EOJ)
	..1.		SIGNTCF	"X'20'" SIGNOFF BY NO REAL SPACE
	.1..		SIGNKILL	"X'40'" STOP LINE WITH KILL FUNCTION
	1...		SIGNLL	"X'80'" STOP LINE AT LAST I/O
(34)	CHARACTER	1	LCBTERR	TERMINAL ERROR COUNT IF COUNT REACHES 10 A SYSREC RECORD IS WRITTEN
(35)	CHARACTER	3	LCBDVAD	LINE ADDRESS 'CUU'
(38)	BITSTRING	1	LCBRMID	REMOTE IDENTIFIER BINARY
(39)	BITSTRING	1	LCBDLCT	DISABLE LOOP COUNTER
	..11 1.1.		LCBSEDA	"*" SESSION DATA
(3A)	SIGNED	2	LCBXCNT	TRANSMISSION COUNT PER SESSION
(3C)	SIGNED	2	LCBTCNT	TIMEOUT COUNT PER SESSION
(3E)	SIGNED	2	LCBECNT	ERROR COUNT PER SESSION
(40)	CHARACTER	6	LCBSFDT	SIGNOFF DATE (MMDDYY)
	.... 11..		LCBSELN	"*-LCBSEDA" LENGTH OF SESSION DATA
	..11 111.		LCBACLG	"*-LCBACCT" LENGTH OF LINE ACCOUNT RECORD
(46)	BITSTRING	1	LCBLREQ	LAST SENT REQUEST
(47)	BITSTRING	1	LCBREQF	REQUEST FIELD FOR I/O MONITOR

Offset Hex	Type	Len	Name (Dim)	Description
(48)	SIGNED	4	LCBNEXT	LCB CHAIN POINTER, LAST = ZERO
(4C)	SIGNED	4	LCBRCPT	POINTER TO NEXT RECORD IN BUFFER
	.1.. 11..		LCBTBRP	"LCBRCPT" TO BE RELEASED POINTER
(50)	SIGNED	2	LCBPUBA	ADDR. OF RELATED PUB ENTRY
(52)	SIGNED	2	LCBLRCL	LENGTH OF LOGICAL RECORD
(54)	SIGNED	2	LCBTIMC	TIME OUT COUNTER This field contains the number of timeouts (1 every 3 seconds) as long as the terminal is idle (no data transfer). When information is transmitted on the line it is set to zero. The count is compared with the timeout limit specified in the PLINE macro.
(56)	BITSTRING	1		LCBRECD: 1st HALFBYTE: RC FOR 1R09I 2nd HALFBYTE: RC FOR 1R07I
(57)	BITSTRING	1	LCBTOCT	TIMEOUTCOUNT FOR PRE-SIGNON AND SWITCHED LINE. IF LIMIT REACHED - FORCE SIGNOFF Count for:- 5 retries of Enable, Nop, Read response CCW sequence if line is in control state before trying to send ENQ. or:- 5 retries of Enable, Nop, Read response CCW sequence if line is in switched mode, not signed on, control state and timed out.
(58)	BITSTRING	1	LCBRTRY	RETRY COUNT FOR UNIT CHECK (MAX 30) Retry count for unit check (max 30). If line is in data transfer mode and received unit check, it is reset at SYSREC writing.
(59)	BITSTRING	1	LCBMSGI	MSG INDEX IN VIRT. MSG QUEUE
(5A)	BITSTRING	1	LCBLIMO	LINE MODE
	1...		LCBMREC	"X'80'" RECEIVE MODE
	.1..		LCBMXMT	"X'40'" TRANSMIT MODE
	..1.		LCBDOUT	"X'20'" DISCONTINUED OUTPUT MODE
	.... 1...		LCBTRDR	"X'08'" RDR TASK INDICATION
	.... .1..		LCBTLST	"X'04'" LST TASK INDICATION
	.... ..1.		LCBTPUN	"X'02'" PUN TASK INDICATION
	1		LCBTMSG	"X'01'" MSG TASK INDICATION
	.... .111		LCBTOUT	"X'07'" OUTPUT TASK INDICATION
(5B)	BITSTRING	1	LCBOUSW	LIST/PUNCH OUTPUT INDICATOR
	1...		LCBOUL1	"X'80'" X'80' LIST OUTPUT QUEUED FOR LST1
	.... 1...		LCBOUP1	"X'08'" X'08' PUNCH OUTPUT
(5C)	BITSTRING	1	LCBACT	ACTIVITY CONTROL BYTE
	1...		LCBATCR	"X'80'" TASK CREATION
	.1..		LCBASHD	"X'40'" SHUTDOWN
	..1.		LCBATSTP	"X'20'" TASK STOP
	...1		LCBASGF	"X'10'" FINAL SIGNOFF
	.... 1...		LCBALSTP	"X'08'" LINE STOP
	.... .1..		LCBALSTR	"X'04'" LINE START
	.... ..1.		LCBALIN	"X'02'" LINE INITIALIZATION
	1		LCBAKILL	"X'01'" LINE STOP (PSTOP CUU,KILL
	1111 1111		LCBAANY	"X'FF'" ANY ACTIVITY FOR THIS LCB
(5D)	BITSTRING	1	LCBTRACE	LINE TRACE INDICATION X'FF'
(5E)	SIGNED	2	LCBNORC	NUMBER OF RECORDS IN BUFFER
(60)	SIGNED	2	LCBMSG#	NUMBER OF REMOTE MESSAGES
(62)	BITSTRING	1	LCBSTMSG	3741-STATUS MSG
(63)	BITSTRING	1	LCBTSKEJ	LAST TASK WHICH SET TURNEOJ
(64)	BITSTRING	24	LCBTQE	SPACE FOR TIMER ELEMENT
(7C)	SIGNED	2		
	PLINE Fields			
(7C)	ADDRESS	2	LCBLPU	PHYSICAL UNIT ADDRESS
(7E)	ADDRESS	2	LCBTLIM	TIME OUT LIMIT(SECONDS) LINE FEATURES
(80)	ADDRESS	1	LCBFEA1	DUAL MODE
	1...		F1ASCII	"X'80'" USASCII
	.1..		F1TRANS	"X'40'" TRANSPARENCY FOR THIS LINE
	..1.		F1SWITC	"X'20'" SWITCHED LINE
(81)	ADDRESS	1	LCBLDM	
	..1.		DUABIFB	"X'20'" BASIC INTERFACE B SELECTED

Offset Hex	Type	Len	Name (Dim)	Description
	.... 1...		DUATRMB	"X'08'" TRANSMISSION MODE B SELECTED
	.... .1..		DUAIRRM	"X'04'" INTRRUPT MODE REQUESTED
	.... .11.		LCBL1	"*-LCBLPU" LENGTH WITHOUT PASSWORD
	PRMT Fields			
	1... ..1.		LCBDRM	"*"
(82)	CHARACTER	3	LCBREMI	REMOTE IDENTIFIER CHAR. FORM
(85)	CHARACTER	2	LCBROUT (0)	ROUTING TARGETS
(85)	ADDRESS	1	LCBPUR	PUNCH ROUTING
(86)	ADDRESS	1	LCBLIR	LIST ROUTING
	1... .111		LCBRCH	"*" CHARACTERISTICS OF REMOTE
(87)	ADDRESS	1	LCBRPD	NUMBER OF PRINT DEVICES
	1... .111		LCBREMX	"LCBRPD" ALSO USED FOR INDICATION
	1111 1111		LCBREME	"X'FF'" ERROR PRMT ENTRY
	1111 111.		LCBREMR	"X'FE'" REFERENCE ENTRY
(88)	ADDRESS	2	LCBBFSZ	REMOTE TERMINAL BUFFER SIZE
(8A)	ADDRESS	1	LCBNRPB	MAX.NR. OF REC'S PER BLOCK
(8B)	ADDRESS	1	LCBTYP	TERMINAL TYPES SUPPORTED
	1... 1.11		LCBREF	"LCBTYP" ALSO USED AS REFERNECE
	....		TYP0	"0" DUMMY
	.... 1.1.		TYP2770	"10" 2770
	...1 .1..		TYP2780	"20" 2780
	...1 111.		TYP3741	"30" 3741
	..11 ..1.		TYP3780	"50" 3780
(8C)	ADDRESS	1	LCBCSAL	COMPONENT SELECT LIST
(8D)	ADDRESS	1	LCBCSAP	COMPONENT SELECT PUNCH
(8E)	ADDRESS	1	LCBCSAM	COMPONENT SELECT MESSAGE
	...1 ...1		FEACSDC1	"X'11'" DC1
	...1 ..1.		FEACSDC2	"X'12'" DC2
	...1 ..11		FEACSDC3	"X'13'" DC3
	.1.1 11.1		FEACS5D	"X'5D'")
	....		FEACS0	"X'00'" NO COMPONENT SELECT
(8F)	BITSTRING	1		RESERVED
(90)	ADDRESS	2	LCB1LLN	1LST PRINT-LENGTH
(92)	ADDRESS	2	LCB1PLN	1PUN
(94)	ADDRESS	2	LCB1MLN	1MSG
	REMOTE FEATURES			
(96)	ADDRESS	1	LCBFEA2	
	..1.		F2LTURN	"X'20'" LINE TURNAROUND BY EOJ
(97)	ADDRESS	1	LCBFEA3	
	.1..		F3TR2H1T	"X'40'" 2-HEADING 1-TRAILING BYTE IN RECORD ON TRANSPARENCY.
(98)	ADDRESS	1	LCBFEA4	
	1...		F4TRANS	"X'80'" TRANSPARENCY FOR THIS REMOTE
	.1..		F4MULRC	"X'40'" MULTIPLE-LOG./PHYS.RECORD (I.E. 3741 EXP.COMMUNICATION)
	..1.		F4HORFC	"X'20'" HORIZONTAL FORMAT CONTROL
	...1		F4BLSCE	"X'10'" SPACE COMPRESSION EXPANSION
	.... 1...		F4COMSL	"X'08'" COMPONENT SELECT
	.... .1..		F4VARLG	"X'04'" VARIABLE LENGTH RECORDC
	.... ..1.		F4NOFC	"X'02'" NO FORMCHANGE SUPPORTED
(99)	ADDRESS	1	LCBFEA5	
	1...		F5TNL	"X'80'" NL CHARACTER ON END OF HT
	..1.		F5EJECT	"X'20'" EJECT BEFORE MESSAGES
	...1		F5SPACE	"X'10'" TERMINAL REQU.SPACE 3 AFTER RECORD I.E. 2770
	.... 1...		F5T3741	"X'08'" TRANSL.TABLE WITH 3741 CTRL
	.... .1..		F5L2780	"X'04'" TERM LIKE 2780
	...1 ..11		LCBRCHL	"*-LCBRCH" CHARACTERISTICS LENGTH
	...1 1...		LCBDRMLN	"*-LCBDRM" REMOTE- ENTRY LENGTH
	1..1 1.1.		LCBNXT	"*" NEXT REMOTE TABLE ENTRY
(9A)	BITSTRING	1	LCBFLG1	LCB FLAG BYTE 1

Offset Hex	Type	Len	Name (Dim)	Description
	1...		LF1CHEND	"X'80'" LCBSCAN CALLED FROM CHEND
	.1..		LF1TERM	"X'40'" TERMINATE SESSION
	..1.		LF1RVIS	"X'20'" RVI SENT
	...1		LF1SIGN	"X'10'" REMOTE SIGNED ON
	.... 1...		LF1SOMQ	"X'08'" SIGNOFF-MSG QUEUED TO REMOTE
	.... .1..		LF1ACWR	"X'04'" ACCOUNT RECORD WRITTEN
	.... ..1.		LF1TOMSG	"X'02'" TIMEOUT MESSAGE QUEUED
	1		LF1EOTR	"X'01'" EOT RECEIVED FOR WRITER
(9B)	BITSTRING	1	LCBFLG2	LCBFLAG BYTE 2
	1...		LF2TIME	"X'80'" TIMER IS SET
	.1..		LF2SETX	"X'40'" ETX TO SEND, NO TASK ACTIVE
	..1.		LF2MSOK	"X'20'" SENDING OF MESSAGES DISABLED
	...1		LF2SENQ	"X'10'" ENQ TO BE SENT
	.... 1...		LF2BUSY	"X'08'" LINE BUSY
	.... .1..		LF2PKMSG	"X'04'" PSTOP KILL MSG QUEUED
	.... ..1.		LF2PMOFF	"X'02'" STOP MSG FOR SIGNOFF
	1		LF2SFRC	"X'01'" FORMS CHANGE IN PROGRESS
(9C)	BITSTRING	1	LCBFLG3	LCBFLAG BYTE 3
	1...		LF3LSCN	"X'80'" TRACE ENTRY FOR ACTIVITY
	.1..		LF3SOFF	"X'40'" SIGNOFF READ BY IPW$$BR
	..1.		LF3MR17	"X'20'" MSG 1R17I ALREADY QUEUED
	...1		LF3EOTR	"X'10'" EOT RECEIVED FOR READER
	.... 1...		LF3EOTS	"X'08'" EOT TO SEND TO RESET TERM
	.... .1..		LF3TCRF	"X'04'" TASK CREATION FAILED
	.... ..1.		LF3EODR	"X'02'" WRITE END-OF-DAY RECORD
	1		LF3TEOJ	"X'01'" TURNAROUND INDICATION
(9D)	BITSTRING	1	LCBRCNT	RETRY COUNT
	1..1 111.		BSCCODE	"*" EBCDIC/USASCII CODE TABLE
(9E)	BITSTRING	2	MSOHSEQ (0)	MULTI-LEAVING SIGN-ON SEQUENCE
(9E)	BITSTRING	1	MBSCSOH	SOH BSC CHARACTER
(9F)	BITSTRING	1	MBSCENQ	ENQ BSC CHARACTER
(A0)	BITSTRING	2	MDLESTX (0)	START OF TEXT SEQUENCE
(A1)	BITSTRING	1	MBSCSTX	STX BSC CHARACTER
(A2)	BITSTRING	2	METBSEQ (0)	END OF TEXT BLOCK SEQUENCE
(A3)	BITSTRING	1	MBSCETB	END OF TEXT BLOCK CHARACTER
(A4)	BITSTRING	2	METXSEQ (0)	END OF TEXT SEQUENCE
(A5)	BITSTRING	1	MBSCETX	ETX BSC CHARACTER
(A6)	BITSTRING	2	MACK0SEQ (0)	EVEN ACKNOWLEDGEMENT SEQUENCE
(A7)	BITSTRING	1	MBSCACK0	EVEN ACKNOWLEDGEMENT CHARACTER
(A8)	BITSTRING	2	MACK1SEQ (0)	ODD ACKNOWLEDGEMENT SEQUENCE
(A9)	BITSTRING	1	MBSCACK1	ODD ACKNOWLEDGEMENT CHARACTER
(AA)	BITSTRING	2	MNAKSEQ (0)	NEGATIVE ACKNOWLEDGEMENT SEQUENCE
(AB)	BITSTRING	1	MBSCNAK	NEGATIVE ACKNOWLEDGEMENT CHARACTER
(AC)	BITSTRING	1	MBSCACKX	ACKNOWLEDGEMENT CONVERSION CHARACTER
(AD)	BITSTRING	1	MBSCCWCH	CCW CHAINING CHARACTER
(AE)	BITSTRING	2	MWACKSEQ (0)	WACK SEQUENCE
(AF)	BITSTRING	1	MBSCWACK	WAIT BEFORE TRANSMIT
(B0)	BITSTRING	2	MEOTSEQ (0)	DLE/EOT SEQUENCE
(B1)	BITSTRING	1	MBSCEOT	EOT BSC CHARACTER
(B2)	BITSTRING	2	MDLEETB	DLE/ETB SEQUENCE
(B4)	BITSTRING	2	MTTDSEQ	STX/ENQ = TTD SEQUENCE
(B6)	BITSTRING	1	MBSCESC	ESCAPE BSC CHARACTER
(B7)	BITSTRING	1	MBSCSYN	SYN BSC CHARACTER
(B8)	BITSTRING	2	MDLEETX	DLE/ETX SEQUENCE
(BA)	BITSTRING	2	MDLERVI	DLE/RVI SEQUENCE
	...1 111.		MBSCLN	"*-BSCCODE"
TABLE OF BSC CONTROL CHARACTERS				
	.... ..1.		MBCDSTX	"X'02'" START OF TEXT
	.... .1.1		MBCDHT	"X'05'" HORIZONTAL TAB
	...1		MBCDDLE	"X'10'" DATA LINK ESCAPE
	...1 .1.1		MBCDNL	"X'15'" NEW LINE

Offset Hex	Type	Len	Name (Dim)	Description
	...1 1..1		MBCDEM	"X'19'" END OF MEDIA
	...1 11.1		MBCDIGS	"X'1D'" INTER-GROUP SEPARATOR
	...1 111.		MBCDIRS	"X'1E'" INTER-RECORD SEPARATOR
	...1 1111		MBCDIUS	"X'1F'" INTER-UNIT SEPARATOR
	..1. .111		MBCDESC	"X'27'" ESCAPE
	..11 ..1.		MBCDSYN	"X'32'" SYNCHRONOUS IDLE
(BC)	BITSTRING	1	LCBFLG4	LCBFLAG BYTE 4
	1...		LF4POUT	"X'80'" OUTPUT TASK TO POST
	.1..		LF4UEXC	"X'40'" UE RECEIVED FOR WRITE ACK
	..1.		LF4LEMSG	"X'20'" LINE ERROR MSG QUEUED
	...1		LF4KPXMT	"X'10'" KEEP TRANSMIT MODE
	.... 1...		LF4VDISC	"X'08'" VALID DISCONNECT RECEIVED
	.... .1..		LF4BYFAB	"X'04'" BYPASS FORWARD ABORT
	.... ..1.		LF4TRANP	"X'02'" REMEMBER TRANSP. PUNCH TASK
	1		LF4DPD	"X'01'" DISABLE PENDING
(BD)	BITSTRING	1	LCBFLG5	LCBFLAG BYTE 5
	1...		LF5TTDR1	"X'80'" TTD RECOVERY INDICATION
	.1..		LF5XXXX2	"X'40'" UNUSED
	..1.		LF5XXXX3	"X'20'" UNUSED
	...1		LF5XXXX4	"X'10'" UNUSED
	.... 1...		LF5XXXX5	"X'08'" UNUSED
	.... .1..		LF5XXXX6	"X'04'" UNUSED
	.... ..1.		LF5XXXX7	"X'02'" UNUSED
	1		LF5XXXX8	"X'01'" UNUSED
(BE)	BITSTRING	1	LCBLRQS	LAST REQUEST SAVED
(BF)	BITSTRING	1		RESERVED

DEFINE BUFFER CONTROL AREA

(C0)	DBL WORD	8		
(C0)	CHARACTER	24	IOBCCB (0)	RJE CCB
(C0)	SIGNED	2	CCBRCNT	RESIDUAL COUNT
(C2)	CHARACTER	1	CCBCOM1	COMMUNICATION BYTE
(C3)	CHARACTER	1	CCBCOM2	COMMUNICATION BYTE
(C4)	SIGNED	2	CCBSTAT	STATUS BYTES FROM CSW
(C6)	SIGNED	2	CCBLUBN	LOGICAL UNIT NUMBER
(C8)	SIGNED	4	IOBSTRT	FIRST CCW ADDRESS
(CC)	CHARACTER	1	CCBCOM3 (0)	COMMUNICATION BYTE
(CC)	SIGNED	4	CCBAPPA	CHANNEL APPENDAGE ADDRESS
(D0)	DBL WORD	8	IOBSCCW	RJE SENSE CCW

RJE CCW String, set up by IPW$$BM

These CCW fields constitute various channel programs that depend upon the operation required. For example, channel end received a data block with correct BSC characters, the response has to be an acknowledgement ACK0/ACK1. IPW$$LM sets up a request in the field LCBREQF, calls IPW$$BM which creates a CCW string based on the request in LCRREQF.

(D8)	DBL WORD	8	IOBCCW1	RJE
(E0)	DBL WORD	8	IOBCCW2	
(E8)	DBL WORD	8	IOBCCW3	CHANNEL
(F0)	DBL WORD	8	IOBCCW4	
(F8)	DBL WORD	8	IOBCCW5	PROGRAM
(100)	DBL WORD	8	IOBCCW6	

Other RJE Information

(108)	SIGNED	4	IOBRSTR	RESTART ADDRESS OF CHANNEL PROGRAM
(10C)	SIGNED	4	IOBNEXT	ADDR OF NEXT I/O BUFFER IN CHANNEL
(110)	SIGNED	4	IOBLCCW	ADDR. OF LAST EXECUTED CCW+8
(114)	SIGNED	2	IOBTACK	ACK TRANSMITTED
(116)	SIGNED	2	IOBEACK	ACK EXPECTED
(118)	SIGNED	4	IOBDISP	DISP BETWEEN REAL AND VIRTUAL
(11C)	CHARACTER	6	IOBRESP (0)	REMOTE RESPONSE CONTROL BLOCK

Offset Hex	Type	Len	Name (Dim)	Description
(11C)	CHARACTER	2	IOBNOID	NO ID FEATURE
(11E)	CHARACTER	2	IOBFILL	WITH IOBNOID = 3741 ID
(120)	CHARACTER	2	IOB3741	RESPONSEFIELD 3741 WITH ID FEATURE
(122)	CHARACTER	1	IOBSNS0	FIRST SENSE BYTE
(123)	CHARACTER	1	IOBSNS1	SECOND SENSE BYTE
(124)	BITSTRING	1	IOBFLG1	IOB FLAG BYTE 1
(125)	BITSTRING	1	IOBFLG2	IOB FLAG BYTE 2
	.11. .11.		IOBLGTH	"*-IOBCCB" LENGTH OF CHANNEL PROG
(126)	BITSTRING	2		UNUSED
DEVICE CONTROL TABLE (DCT) WITHIN LCB: OUTPUT DCT'S				
List DCT				
(128)	SIGNED	4	DCT1LST (0)	1LST DCT SECTION
(128)	CHARACTER	4	DCT1LID	LST TASK IDENTIFIER
(12C)	ADDRESS	4	DCT1LTCB	TCB ADDRESS OF TASK
(130)	BITSTRING	1	DCT1LST1	STATUS BYTE 1 (see Reader DCT)
(131)	BITSTRING	1	DCT1LST2	STATUS BYTE 2 (see Reader DCT)
(132)	BITSTRING	1	DCT1LST3	STATUS BYTE 3 (see Reader DCT)
(133)	BITSTRING	3		UNUSED
(136)	BITSTRING	1	DCT1LFLG	FLAG BYTE
	.1..		DCTPMSG	"X'40'" STOP ACT.MSG TASK DUE TO COMMAND
	..1.		DCTSKIP	"X'20'" SKIP TO CH1 INSERTION REQ.
	...1		DCTSIGN	"X'10'" SHOW IGN. RECORDS REQUESTED
(137)	BITSTRING	1	DCT1LIDI	HEX IDENTIFIER(AS LCBOUSW)
(138)	BITSTRING	4	DCT1LCLS	LST CLASSES
(13C)	SIGNED	2	DCT1LPLN	LENGTH OF PRINT LINE
(13E)	BITSTRING	1	DCT1LNRC	NUMBER OF REC/BUFFER
(13F)	BITSTRING	1	DCT1LIDH	HEX IDENTIFIER(INV.LCBOUSW)
(140)	BITSTRING	1	DCT1LSCT	SPACE 3 LINE COUNT
(141)	BITSTRING	1	DCT1LCSC	COMPONENT SELECT CHAR
(142)	BITSTRING	4	DCT1LFRM	FORMS IDENTIFIER
(146)	BITSTRING	1	DCT1LPRC	PREVIOUS COMMAND CODE
(147)	BITSTRING	1	DCT1LBST	BUFFER STATUS FLAGS
(148)	SIGNED	4		FW-BOUNDARY
	..1.		DCT1LLI	"*-DCT1LID" LENGTH OF 1LST DCT
Punch DCT				
(148)	SIGNED	4	DCT1PUN (0)	1PUN DCT SECTION
(148)	CHARACTER	4	DCT1PID	PUN TASK IDENTIFIER
(14C)	ADDRESS	4	DCT1PTCB	TCB ADDRESS OF TASK
(150)	BITSTRING	1	DCT1PST1	STATUS BYTE 1 (see Reader DCT)
(151)	BITSTRING	1	DCT1PST2	STATUS BYTE 2 (see Reader DCT)
(152)	BITSTRING	1	DCT1PST3	STATUS BYTE 3 (see Reader DCT)
(153)	BITSTRING	3		UNUSED
(156)	BITSTRING	1	DCT1PFLG	FLAG BYTE
(157)	BITSTRING	1	DCT1PIDI	HEX IDENTIFIER(AS LCBOUSW)
(158)	BITSTRING	4	DCT1PCLS	LST CLASSES
(15C)	SIGNED	2	DCT1PPLN	LENGTH OF PRINT LINE
(15E)	BITSTRING	1	DCT1PNRC	NUMBER OF REC/BUFFER
(15F)	BITSTRING	1	DCT1PIDH	HEX IDENTIFIER(INV.LCBOUSW)
(160)	BITSTRING	1	DCT1PSCT	SPACE 3 LINE COUNT
(161)	BITSTRING	1	DCT1PCSC	COMPONENT SELECT CHAR
(162)	BITSTRING	4	DCT1PFRM	FORMS IDENTIFIER
(166)	BITSTRING	1	DCT1PPRC	PREVIOUS COMMAND CODE
(167)	BITSTRING	1	DCT1PBST	BUFFER STATUS FLAGS
(168)	SIGNED	4		FW-BOUNDARY
	..1.		DCT1PLI	"*-DCT1PID" LENGTH OF 1PUN DCT
Message Task DCT				

Offset Hex	Type	Len	Name (Dim)	Description
(168)	SIGNED	4	DCT1MSG (0)	1MSG DCT SECTION
(168)	CHAR-ACTER	4	DCT1MID	MSG TASK IDENTIFIER
(16C)	ADDRESS	4	DCT1MTCB	TCB ADDRESS OF TASK
(170)	BITSTRING	1	DCT1MST1	STATUS BYTE 1 (see Reader DCT)
(171)	BITSTRING	1	DCT1MST2	STATUS BYTE 2 (see Reader DCT)
(172)	BITSTRING	1	DCT1MST3	STATUS BYTE 3 (see Reader DCT)
(173)	BITSTRING	3		UNUSED
(176)	BITSTRING	1	DCT1MFLG	FLAG BYTE
(177)	BITSTRING	1		RESERVED
(178)	BITSTRING	4	DCT1MCLS	NOT APPLICABLE
(17C)	SIGNED	2	DCT1MPLN	LENGTH OF PRINT LINE
(17E)	BITSTRING	1	DCT1MNRC	NUMBER OF REC/BUFFER
(17F)	BITSTRING	1	DCT1MIDH	NOT APPLICABLE
(180)	BITSTRING	1	DCT1MSCT	SPACE 3 LINE COUNT
(181)	BITSTRING	1	DCT1MCSC	COMPONENT SELECT CHAR
(182)	BITSTRING	4	DCT1MFRM	NOT APPLICABLE
(186)	BITSTRING	1	DCT1MPRC	PREVIOUS COMMAND CODE
(187)	BITSTRING	1	DCT1MBST	BUFFER STATUS FLAGS
(188)	SIGNED	4		FW-BOUNDARY
	..1.		DCT1MLI	"*-DCT1MID" LENGTH OF 1MSG DCT
	11		DCTLCTR	"3" NUMBER OF OUTPUT DCT'S
	Reader DCT			
(188)	SIGNED	4	DCT1RDR (0)	1RDR DCT SECTION
(188)	CHAR-ACTER	4	DCT1RID	RDR TASK IDENTIFIER
(18C)	ADDRESS	4	DCT1RTCB	TCB ADDRESS OF TASK
(190)	BITSTRING	1	DCT1RST1	STATUS BYTE 1
	1...		DCTSOLI	"X'80'" LOGICAL INTERFACE OPENED
	.1..		DCTLERR	"X'40'" WRONG RECORD LENGTH
	..1.		DCTINTB	"X'20'" BUFFER TO INITIALIZE
	...1		DCTINIT	"X'10'" START COMMAND RECEIVED
	.... 1...		DCTSDET	"X'08'" DETACH REQUESTED
	.... .1..		DCTEOFR	"X'04'" END OF FILE RECEIVED ON RDR
	.... ..1.		DCTSENQ	"X'02'" TASK IS READY TO SENT
	1		DCTSTRT	"X'01'" START THIS TASK
(191)	BITSTRING	1	DCT1RST2	STATUS BYTE 2
	1...		DCTEOFD	"X'80'" END-OF-FILE DETECTED
	.1..		DCTEOJD	"X'40'" END-OF-JOB DETECTED
	..1.		DCTOPDO	"X'20'" OPEN PROCEDURE PERFORMED
	...1		DCTMSIP	"X'10'" MESSAGE IN PROCESS
	.... 1...		DCTMSGT	"X'08'" TEMPORARY MESSAGE TASK
	.... .1..		DCTSFRC	"X'04'" FORMS CHANGE NEEDED
	.... ..1.		DCTEOBD	"X'02'" END OF BLOCK DETECTED
	1		DCTMFRSI	"X'01'" STOP IMM IN MOUNT FORMS
(192)	BITSTRING	1	DCT1RST3	STATUS BYTE 3
	1...		DCTLCSP	"X'80'" LAST COMMAND WAS SPACE
	.1..		DCTLCEJ	"X'40'" LAST COMMAND WAS AN EJECT
	..1.		DCTCSIP	"X'20'" COMPONENT SELECT IN PROCESS
	...1		DCTTRANS	"X'10'" TRANSPARENCY FOR THIS TASK
	.... 1...		DCT1R19I	"X'08'" MSG 1R19I ALREADY SENT (BR)
	.... .1..		DCTCSF	"X'04'" COMPONENT SELECT FAILED
	.... ..1.		DCTMDEL	"X'02'" MSG TO DELETE
	1		DCTSETU	"X'01'" SETUP COMMAND PROCESSING
(193)	BITSTRING	1		RESERVED
(194)	BITSTRING	1	DCT1RCLS	RDR CLASS
(195)	BITSTRING	3		UNUSED
(198)	SIGNED	4		FW-BOUNDARY
	...1		DCT1RLI	"*-DCT1RID" LENGTH OF 1RDR DCT
(198)	SIGNED	4		ALIGN TO FULLWORD BOUNDARY
	TRANSMISSION BUFFERS + Work Areas			
(198)	CHAR-ACTER	528	LCBRECBF (0)	DEFINE RECEIVE BUFFER

Offset Hex	Type	Len	Name (Dim)	Description
(198)	SIGNED	4	LCBRBFAR	REAL ADDRESS OF RECEIVE BUFF
(19C)	BITSTRING	1	LCBRBFST	RECEIVE BUFFER STATUS BYTE
	1...		LCBBUSE	"X'80'" BUFFER IN USE
	.1..		LCBTDET	"X'40'" TASK TO BE DETACHED
	..1.		LCBBSETX	"X'20'" ETX SENT
	...1		LCBBREOT	"X'10'" EOT RECEIVED
(19D)	BITSTRING	1	LCBRRFLD	RECEIVE BUFFER REQUEST FIELD
(19E)	SIGNED	2	LCBRDATL	RECEIVE BUFFER DATA LENGTH
(1A0)	SIGNED	4	LCBREOBA	RECEIVE BUFFER END ADDRESS
(1A4)	CHARACTER	516	LCBRBUF	RECEIVE BUFFER
(3A8)	CHARACTER	532	LCBWRTBF (0)	DEFINE TRANSMIT BUFFER
(3A8)	SIGNED	4	LCBWBFAR	REAL ADDRESS TRANSMIT BUFFER
(3AC)	BITSTRING	1	LCBWBFST	TRANSMIT BUFFER STATUS BYTE (see LCBRBFST for definitions)
(3AD)	BITSTRING	1	LCBWRFLD	TRANSMIT BUFFER REQU FIELD
(3AE)	SIGNED	2	LCBWDATL	TRANSMIT BUFFER DATA LENGTH
(3B0)	SIGNED	4	LCBWEOBA	TRANSMIT BUFFER CURR. RECORD
(3B4)	CHARACTER	520	LCBWBUF	TRANSMIT BUFFER
(5BC)	CHARACTER	186	LCBLWAW (0)	WORKAREA FOR WRITER
(5BC)	SIGNED	4	LCBLCCA	LAST RECORD CARR.CTRL ADDR.
(5C0)	SIGNED	2	LCBWADL	DATA LENGTH IN WORK AREA
(5C2)	BITSTRING	1	LCBWACE	ESCAPE CHARACTER FOR LISTOUT
(5C3)	BITSTRING	1	LCBWACC	CARRIAGE CONTROL FOR LISTOUT
(5C4)	CHARACTER	178	LCBWADA	DATA AREA
	1.11 .1..		LCBWALG	"*-LCBWACE" LENGTH OF WORK AREA USED
(676)	CHARACTER	200	LCBLWAR	WORKAREA FOR READER
	EXPRESSION		LCBTLG	"*-LCBHEAD" TOTAL LENGTH OF LCB

Segment Macro Parameter List

Definition Macro: IPW$MXD

The segment macro parameter list is generated by the macro IPW$MXD for use by the user when calling the IPWSEGM macro. It creates a workarea for storing data and a CCB/CCW which will be created in the workarea when the IPWSEGM processing is completed by code in IPW$$NU.

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
					Segment Macro Parameter List
(0)	0	DBL WORD	8		REFLECT ACTUAL DOUBLE WORD ALIGNMENT
(0)	0	BITSTRING	16	$MXCCB	COMMAND CONTROL BLOCK
(10)	16	DBL WORD	8	$MXCCW	CHANNEL COMMAND WORD
(18)	24	SIGNED	4	$MXRSV (13)	SAVE AREA REGISTER 2 - 14
(4C)	76	BITSTRING	9	$MXDJA	DEFAULT JECL STATEMENT AREA
(55)	85	BITSTRING	3		UNUSED
(58)	88	BITSTRING	24	$MXINP (0)	INPUT AREA TO VSE/POWER
(58)	88	BITSTRING	4	$MXVRS	VERSION OF PARAMETER AREA
(5C)	92	BITSTRING	4	$MXUNA (0)	LOGICAL UNIT ADDRESS
(5C)	92	BITSTRING	1	$MXCLS	LOGICAL UNIT CLASS
(5D)	93	BITSTRING	1	$MXNUM	LOGICAL UNIT NUMBER
(5E)	94	BITSTRING	2		LOG. UNIT ADDRESS BYTE 2+3
(60)	96	BITSTRING	4	$MXJCL	ADDRESS OF JECL STATEMENT
(64)	100	BITSTRING	4	$MXJCN	LENGTH OF JECL STATEMENT
(68)	104	BITSTRING	1	$MXOP1	INPUT OPTION BYTE 1
		1...		$MX1UA	"X'80'" .. LOG. UNIT BY ADDRESS
		.1..		$MX1PJ	"X'40'" .. PASSED JECL OF USER
		..1.		$MX1KP	"X'20'" .. KEEP OPTION SPECIFIED
(69)	105	BITSTRING	7	$MXRDI	RESERVED INPUT AREA
(70)	112	BITSTRING	28	$MXRET (0)	RETURN AREA TO USER PROGRAM
(70)	112	CHARACTER	1	$MXSQI	QUEUE-ID OF CREATED SEGMENT (R|L|P|X)
(71)	113	CHARACTER	1	$MXSCL	JOB CLASS OF CREATED SEGMENT (0-9,A-Z)
(72)	114	BITSTRING	1	$MXJSF	OUTPUT SUFFIX, IF 'RBS=' USED
		1...		$MXJSFL	"X'80'" .. 'LAST RBS SEGMENT' FLAG
					X'7F' .. RBS SEGMENT SUFFIX NUMBER (BINARY)
(73)	115	BITSTRING	1		UNUSED
(74)	116	CHARACTER	8	$MXJNM	JOB NAME OF CREATED SEGMENT
(7C)	124	BITSTRING	4	$MXJNB	JOB NUMBER OF CREATED SEGMENT (BINARY)
(80)	128	BITSTRING	4	$MXQNB	BIN. Q-ENTRY NUMBER OF CREATED SEGMENT
(84)	132	BITSTRING	2	$MXRRF (0)	REGISTER 15 RETURN/FEEDBACK CODES
(84)	132	BITSTRING	1	$MXRRC	RETURN CODE
		....		$MXR00	"X'00'" .. OK, NO ERROR (PERHAPS WARNING)
		.... .1..		$MXR04	"X'04'" .. INITIALIZATION ERROR
		.... 1...		$MXR08	"X'08'" .. SPECIFICATION INCONSISTENCIES
		.... 11..		$MXR0C	"X'0C'" .. EXECUTION PROCESSING ERROR
(85)	133	BITSTRING	1	$MXRFB	FEEDBACK CODE
		....		$MX00OK	"X'00'" .. OK
		.... .1..		$MX00IG	"X'04'" .. NOTHING SPOOLED
					X'08' .. UNUSED
		.... 11..		$MX00PU	"X'0C'" .. OUTPUT PURGED
		...1		$MX00NK	"X'10'" .. DISP=N OK, SPOOLING STOPS
		...1 .1..		$MX00NE	"X'14'" .. DISP=N ERROR, SET DISP=D
		.... .1..		$MX04PNA	"X'04'" .. VSE/POWER NOT ACTIVE
		.... .1..		$MX08NPC	"X'04'" .. PARTITION NOT POWER CONTROLLED
		.... 1...		$MX08NSY	"X'08'" .. DEVADDR NOT STARTING 'SYS...'
		.... 11..		$MX08ILU	"X'0C'" .. INCORRECT LOGICAL UNIT 'SYSXXX', NEITHER 'XXX' = 000-255 NOR 'XXX' = PCH|LST
		...1		$MX08IPD	"X'10'" .. INVALID PUB DEVICE FOR 'SYSXXX', NEITHER PRINTER NOR PUNCH TYPE

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
		...1 .1..		$MX08NPS	"X'14'" .. 'SYSXXX' NO POWER SPOOLED DEVICE
		...1 1...		$MX08UNA	"X'18'" .. 'SYSXXX' UNASSIGNED OR IGNORE
		...1 11..		$MX08IVR	"X'1C'" .. 'SYSXXX' INTERNAL ERROR, CALL IBM
		..1.		$MX08CDN	"X'20'" .. 'SYSXXX' CURRENTLY DISP=N SPOOLED
		..1. .1..		$MX08PWW	"X'24'" .. PARTITION JUST 'WAITING FOR WORK', WITH NO VSE/POWER JOB ACTIVE
		..1. 1...		$MX08IJL	"X'28'" .. INCORRECT JECL LENGTH, JECLN NOT WITHIN LIMITS 9 - 1024
		..1. 11..		$MX08IJS	"X'2C'" .. INCORRECT JECL STATEMENT, NOT STARTING ' $$ LST ' OR ' $$ PUN '
		..11		$MX08NMD	"X'30'" .. NO MATCHING DEVICE TYPE OF 'SYSXXX' VERSUS ' $$ LST/PUN'
		..11 .1..		$MX08FCD	"X'34'" .. CDLOAD 3800-IJDANCHX FAILS DUE TO RESOURCE SHORTAGE
		..11 1...		$MX08PNF	"X'38'" .. CDLOAD 3800-IJDANCHX FAILS DUE TO PHASE NOT FOUND
		..11 11..		$MX08UGF	"X'3C'" .. 'GETFLD' UNEXPECTED RETURN CODE
		.1..		$MX08UCD	"X'40'" .. 'CDLOAD' UNEXPECTED RETURN CODE
		.1.. .1..		$MX08CSP	"X'44'" .. CONTRADICTION 'GETFLD' VERSUS DEVICE ENTRY SCAN, CALL IBM

NOTE IBM - MOST OF THE FOLLOWING ARE RETURN CODES FROM
IPW$$XJ. BE SURE TO CO-ORDINATE ANY CHANGES.

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
		.... .1..		$MX0CNOM	"X'04'" .. NO MATCHING SPOOL DEVICE
		.... 1...		$MX0CDEL	"X'08'" .. INVALID OPERAND DELIMITER
		.... 11..		$MX0CUNK	"X'0C'" .. UNKNOWN KEYWORD
		...1		$MX0CINV	"X'10'" .. INVALID OPERAND VALUE
		...1 .1..		$MX0CSTP	"X'14'" .. OPERATOR CANCELLED TAPE
		...1 1...		$MX0CINE	"X'18'" .. INTERNAL POWER ERROR
		...1 11..		$MX0CINA	"X'1C'" .. INVALID 'JECL' ADDRESS (FOLLOWING ARE IBM INTERNAL)
		..1.		$MX0CINS	"X'20'" .. RESERVED (FOR IPW$$XJ)
		..1. .1..		$MX0CIGN	"X'24'" .. RESERVED (FOR IPW$$XJ)
		..1. 1...		$MX0COPF	"X'28'" .. RESERVED (FOR IPW$$XJ)
		..1. 11..		$MX0CUNJ	"X'2C'" .. RESERVED (FOR IPW$$XJ)
		..11		$MX0CSEG	"X'30'" .. RESERVED (FOR IPW$$XJ)
		..11 .1..		$MX0CSEM	"X'34'" .. RESERVED (FOR IPW$$XJ)
(86)	134	BITSTRING	6	$MXRDR	RESERVED RETURN AREA
(8C)	140	BITSTRING	8	$MXALN	DOUBLE WORD ALIGNMENT BUFFER
		1... 11..		$MXSEGMI	"X'8C'" VSE/POWER CAT OFFSET TO 'CASEGMI'
		1..1 .1..		$MXLEN	"*-$MXDS" LENGTH OF PARAMETER AREA (MAX 256) ASM H V 02 12.45

Shared System Slot Communication (SLOT)

Definition Macro: IPW$DEF SLOT=YES

Shared VSE/POWER systems communicate via slots, whose layouts are below.

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
SLOT LAYOUT					
(0)	0	STRUCTURE	0	SLOTDS	, SLOT LAYOUT
(0)	0	CHARACTER	4	SLOTHDR (0)	SLOT HEADER
(0)	0	BITSTRING	1	SLOTSID	SYSTEM IDENTIFIER
(1)	1	BITSTRING	1	SLOTTYPE	SLOT TYPE
		1		SLOTTWFW	"X'01'" .. WAITING FOR WORK SLOT
		.... ..1.		SLOTTNMR	"X'02'" .. MESSAGE/COMMAND SLOT
		11		SLOTTCKP	"X'03'" .. CHECKPOINT SLOT
(2)	2	ADDRESS	2	SLOTBLEN	LENGTH OF SLOT BODY
(4)	4	CHARACTER	1	SLOTBODY (0)	BEGIN OF SLOT BODY

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
WAITING-FOR-WORK SLOT BODY					
(0)	0	STRUCTURE	0	WFWSDS	, WAITING FOR WORK SLOT BODY
(0)	0	CHARACTER	8	WFWSDEV	DEVICE NAME
(8)	8	CHARACTER	4	WFWSCLS	CLASS(ES)
(C)	12	CHARACTER	1	WFWSTYPE	QUEUE TYPE TO BE PROCESSED
(D)	13	BITSTRING	1	WFWSFLG	NOTIFICATION FLAG BYTE
		1...		WFWSFLVE	"X'80'" .. OUTPUT AVAILABLE
(E)	14	CHARACTER	8	WFWSDEST (8)	LOG. DESTINATION NAMES
		.1.. 111.		WFWSLEN	"*-WFWSDS" LENGTH OF WFW SLOT BODY

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
CHECKPOINT SLOT BODY					
(0)	0	STRUCTURE	0	SLOCKPDS	, CHECKPOINT SLOT BODY
(0)	0	SIGNED	2	SLOCKPLN	LENGTH OF BODY + VARIABLE
(2)	2	BITSTRING	1	SLOCKPQI	QUEUE ID
(3)	3	BITSTRING	1	SLOCKPFG	FLAG BYTE 1
		1...		SLOCKPFD	"X'80'" .. SLOT TO BE DELETED
		.1..		SLOCKPFR	"X'40'" .. SLOT RECOVERY TRIED
(4)	4	CHARACTER	8	SLOCKPJN	JOBNAME
(C)	12	SIGNED	2	SLOCKPJO	JOBNUMBER
(E)	14	BITSTRING	1	SLOCKPJS	JOBSUFFIX
(F)	15	BITSTRING	1	SLOCKPCP	COPY NUMBER
(10)	16	SIGNED	4	SLOCKPRN	RECORD NUMBER
(14)	20	SIGNED	4	SLOCKPQN	QUEUE RECORD NUMBER
(18)	24	SIGNED	4		RESERVED FOR FUTURE USE
		...1 11..		SLOCKPFL	"*-SLOCKPDS" LENGTH OF CKP FIXED PART

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(1C)	28	CHAR-ACTER	1	SLOCKPXI (0)	START OF EXTENDED CKP INFO LENGTH IS VARIABLE !

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
				SLOT DBLK LAYOUT	
(0)	0	STRUC-TURE	0	SDBLKDS	, LAYOUT OF SLOT-DBLK
(0)	0	CHAR-ACTER	32	SDBHSER	FIRST PART OF SER RECORD
(20)	32	ADDRESS	4	SDBHPREV	ADDRESS OF PREVIOUS DBLK
(24)	36	ADDRESS	4	SDBHNEXT	ADDRESS OF NEXT SLOT DBLK
(28)	40	BITSTRING	1	SDBHFLAG	FLAG BYTE
		1...		SDBHF1ST	"X'80'" ..FIRST DBLK IN DBLK GROUP
		.1..		SDBHFNXT	"X'40'" ..NEXT DBLK PRESENT
		..1.		SDBHFPRV	"X'20'" ..PREV DBLK IN CHAIN EXISTS
(29)	41	BITSTRING	1		RESERVED FOR FUTURE
(2A)	42	ADDRESS	2	SDBHDISP	OFFSET TO FREE SPACE IN DBLK
(2C)	44	ADDRESS	2	SDBHNOFB	NUMBER OF FREE BYTES IN DBLK
(2E)	46	ADDRESS	2	SDBHBUCK (11)	NO OF SLOTS/SYSID SYSID=10 FOR CKP SLOTS
(44)	68	CHAR-ACTER	1	SDBLKSLO (0)	BEGIN OF SLOT ENTRIES

Service Request Block (SRB)

Definition Macro: IPW$DSR

A service request block is created whenever a service request is passed to asynchronous service for processing.

During the time asynchronous service is performing the service request, the SRBs are chained together.

Asynchronous service handles the request on a 'first-in, first-out' basis.

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
					SERVICE REQUEST BLOCK (SRB)
(0)	0	ADDRESS	4	SRBNEXT	PTR TO NEXT SRB IN CHAIN
(4)	4	BITSTRING	1	SRBREQ	REQUEST CODE
		CHARACTER		SRBFCB	"C'B'" .. LOAD FCB REQUEST
		CHARACTER		SRBFND	"C'C'" .. FIND MEMBER REQUEST D03PILR
		CHARACTER		SRBDIS	"C'D'" .. DISCONNECT MEMBER REQUEST
		CHARACTER		SRBFEOV	"C'E'" .. CALL BAM FEOV REQUEST
		CHARACTER		SRBFT	"C'F'" .. CALL TRANSIENT REQUEST
		CHARACTER		SRBGVCE	"C'G'" .. CALL TRANSIENT REQUEST
		CHARACTER		SRBIDR	"C'I'" .. IDUMP REQUEST
		CHARACTER		SRBLD	"C'L' .. LOAD REQUEST
		CHARACTER		SRBFNDV	"C'M'" .. FIND MEM NON-PWR LIBDEF (LBRACCES BUILD CHAIN)
		CHARACTER		SRBNTE	"C'N'" .. 'NOTE' REQUEST
		CHARACTER		SRBOPN	"C'O'" .. OPEN ICCF INTERFACE
		CHARACTER		SRBPNT	"C'P'" .. 'POINT' REQUEST
		CHARACTER		SRBGRC	"C'R'" .. GET RECORD REQUEST
		CHARACTER		SRBSPR	"C'S'" .. SETPRT REQUEST
		CHARACTER		SRBDISV	"C'T'" .. DISC MEM NON-PWR LIBDEF
		CHARACTER		SRBLDV	"C'V'" .. GET SVA ENTRY POINT
		CHARACTER		SRBFNDV1	"C'W'" .. FIND MEM NON-PWR LIBDEF (INLMFIND FIND MEMBER)
		CHARACTER		SRBIJBX	"C'X'" .. INVOKE IJBXPCA FOR XECB
		CHARACTER		SRBLDY	"C'Y'" .. LOAD DYNAMIC CLASS TABLE
(5)	5	BITSTRING	1	SRBRTC	RETURN CODE
		1111 111.		SRBRFE	"X'FE'" .. IJBXPCA NOT AVAILABLE
		1111 1111		SRBRFF	"X'FF'" .. REQUEST FAILED, TERMIN.
		1...		SRBPSVA	"X'80'" .. PHASE LOCATED IN SVA
		.... 1...		SRBPTL	"X'08'" .. PHASE TOO LARGE
		.... .1..		SRBPNF	"X'04'" .. PHASE NOT FOUND
		.... ..1.		SRBATCAN	"X'02'" .. CANCEL PASSED VIA $AT
		1		SRBATLIO	"X'01'" .. LIOCS FAILED
(6)	6	BITSTRING	1	SRBRTCAF	AF SECHECK MACRO RET CODE (SEE SGACF MACRO)

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(7)	7	BITSTRING	1	SRBOPT	OPTION BYTE
		1...		SRBCSVA	"X'80'" .. CHECK IF PHASE IN SVA
(8)	8	SIGNED	4	SRBECB	EVENT CONTROL BLOCK
(C)	12	CHAR-ACTER	2	SRBPID	PARTITION ID OF REQUESTOR
(E)	14	BITSTRING	1	SRBFLG	SRB FLAG BYTE
		1...		SRBFATF	"X'80'" .. ATTACH DUMP SUBT. FAILED
(F)	15	BITSTRING	1		UNUSED
SETPRT Request					
(10)	16	SIGNED	4	SRBPARM (4)	REQUEST PARAMETER LIST
THE FOLLOWING RE-DEFINITIONS ARE USED FOR THE LOAD REQUEST					
(10)	16	ADDRESS	4	SRBAPN	ADDRESS OF PHASE NAME
(14)	20	ADDRESS	4	SRBLDA	ADDR WHERE TO LOAD PHASE
(18)	24	SIGNED	4	SRBPSZ	LENGTH OF PHASE IN BYTES
(18)	24	SIGNED	4	SRBABC	ADDRESS OF CCB FOR LFCB REQUEST
(1C)	28	ADDRESS	4	SRBEDA	PHASE ENTRY POINT ADDR
THE FOLLOWING RE-DEFINITIONS ARE USED FOR IDUMP SERVICE					
(10)	16	ADDRESS	4	SRBTSA	TRACE START ADDRESS
(14)	20	ADDRESS	4	SRBTEA	TRACE END ADDRESS
THE FOLLOWING RE-DEF'S ARE USED FOR DYNCLASS ID=YES					
(10)	16	ADDRESS	4	SRBCLS	ADDR. OF CLASS TABLE AREA
(14)	20	ADDRESS	4	SRBCRF	DYNCLASS ID=GET RF RET.CODE
(18)	24	ADDRESS	4	SRBCR0	DYNCLASS ID=GET R0 RET.CODE
THE FOLLOWING REDEFINITIONS ARE USED FOR SLI SUPPORT					
(10)	16	ADDRESS	4	SRBSLWA	ADDRESS OF SL WORK AREA
(14)	20	ADDRESS	4	SRBSLME	ADDR OF SL MEMBER ELEMENT
(18)	24	ADDRESS	4	SRBSLPD	ADDR OF PARTITION CNTRL BLK
THE FOLLOWING REDEFINITIONS ARE USED FOR GETVCE					
(10)	16	ADDRESS	4	SRBAVR	POINTER TO AVR LIST
(14)	20	BITSTRING	2	SRBPHLU	LOGICAL UNIT
(16)	22	BITSTRING	1	SRBMACID	SERVICE MACRO ID
THE FOLLOWING REDEFINITIONS ARE USED FOR IJBXPCA					
(10)	16	BITSTRING	4	SRBXU1	XECB USERID BYTES 0-3
(14)	20	BITSTRING	4	SRBXU2	XECB USERID BYTES 4-7
(18)	24	BITSTRING	1	SRBXFG	SECURITY FLAG
MISCELLANEOUS SRBRTC AND SRBRTCF ARE BOTH FILLED ON RETURN FROM SETPRT (IPW$$AS) WHEREBY SRBRTC AND SRBRTCF+3 ARE IDENTICAL. USUALLY SRBRTC WOULD BE A REDEFINITION OF SRBRTCF BUT HAS NOT BEEN CHANGED TO MAINTAIN COMPATIBILITY TO OTHER SOFTWARE (OEM).					
(20)	32	BITSTRING	4	SRBRTCF	FOUR BYTE RETURN CODE FROM SETPRT
(24)	36	BITSTRING	2		UNUSED
		..1. .!!.		SRBLN	"*-SRBDS" LENGTH OF CONTROL BLOCK ASM H V 02 19.22

SNA Session Control Block for PNET (SSCB)

Definition Macro: IPW$DSS

A SNA session control block is created is a SNA session is established to another node in the network. The SSCB is anchored to the appropriate node control block (NCB).

Bytes Hex.	Label of Field	Description/Function
00-0F	SSCBSD	Storage descriptor
• Function Save Areas		
10-3F	SSCBFUSS	Function save area for SEND exit
40-6F	SSCBFUSR	Function save area for RECEIVE exit
• Save Areas for VTAM Macro Calls		
70-73	SSCBLS13	Reg 13 save area on SEND, PNET Driver
74-BB	SSCBVSS1	Save area used for SEND macro
BC-BF	SSCBES13	Reg 13 save area on SEND, SEND EXIT
C0-107	SSCBVSS2	Save area used for CHECK/SEND, SEND EXIT
108-10B	SSCBLR13	Caller's reg 13 save area if RECEIVE
10C-153	SSCBVSR1	Save area used for RECEIVE macro
154-157	SSCBER13	Caller's reg 13 save area, RECEIVE EXIT
158-19F	SSCBVSR2	Save area used for RECEIVE macro, RECEIVE exit
1A0-1E7	SSCBVDSA	Save area used by disconnect task
1E8-1EB	SSCBR14S	VTAM return address, SEND exit
1EC-1EF	SSCBR14R	VTAM return address, RECEIVE exit
• Save Area for Individual Addresses/Pointers		
1F0-1F3	SSCBSRQE	Connect SRQE address
1F4-1F7	SSCBNCB	Connect Node control block address
1F8-1FB	SSCBS2P	Connect save area PAP
1FC-1FF	SSCBS2S	Connect save area SAP
200-203	SSCBS3	Disconnect save area
204-267	SSCBSRPL	RPL skeleton
268-2A7	SSCBNIB	NIB skeleton
• BIND Image and FM-Headers		
2A8-2CB	SSCBBIND	Bind RU
2CC-2D3	SSCBFMHO	FM header (output)
2D4-2DB	SSCBFMHI	FM header (input)

SNA Compaction Table Control Block (COCB)

Definition Macro: IPW$DCO

The address (COAD) in a COCB entry will be used for retrieving the FMH3, and fetching the compaction table for use in the compaction algorithm.

The format of this block is as follows:

Bytes Hex.	Label of Field	Description/Function
00-0F	COSD	Storage descriptor (COCB)
10-13	CONX	Address next COCB
14-15	CONE	Number of entries in COCB
16-17	COTG	Maximum number of GETVIS table entries (1K each)
18-19	COAG	Actual number of GETVIS
1A-1F		Reserved
20-2F	First Compaction Table Entry	
20-23	CONA	Compaction table name
24-27	COAD	Compaction table address
28	COID	Compaction table identifier
29		Reserved
2A-2B	COUS	Compaction table use counter
2C-2D	COLN	Compaction table length
2E-2F		Reserved
30-3FF	Remaining Table Entries	

SNA Control Block (SNCB)

Definition Macro: IPW$DSN

The SNA control block contains general information that is required in real storage for RJE,SNA processing.

Bytes	Label	
Hex.	of Field	Description/Function
00-0F	SNSD	Storage descriptor (SNCB)
10	SNTT	SNA termination type
11	SNTX	Termination type set by SNA exit routines
12	SNFL	Flag byte:
	SNSS	X'80' - SNA stop requested
	SNKS	X'40' - Kill SNA requested
	SNST	X'20' - Subtask detach requested
	SNRQ	X'10' - Subtask quiesce requested
	SNTPN	X'08' - TPEND Exit Driven - Normal
	SNTPQ	X'04' - TPEND Exit Driven - Quick
	SNTPA	X'02' - TPEND Exit Driven - Abend
	SNVA	X'01' - VTAM abended
13	SNSU	Maximum number of logical units
14-17	SNFS	Address of first active SNA unit control block (SUCB)
18-1B	SNTC	Address of SNA manager TCB
1C-1F	SNLW	SNA control block lockword
20-23	SNRM	Address of SNA remote control block (RMCB)
24-27	SNRL	Lockword for general purpose work space (RMGP) in SNA remote control block
28-2B	SNSB	Subtask ECB
2C-2F	SNEB	SNA manager work ECB
2C-2D		Unused
2E		Post byte
	SNEP	X'80' - post bit
2F		Unused
30-37		Reserved for future use
38-3B	SNLR	Address first logon request control block (LRCB)
3C-3F	SNWS	Address logon SUCB
40-43	SNCA	Address of compaction table
44-47	SNEC	LRCB chain - lockword 1
48-4B	SNFC	LRCB chain - lockword 2
4C-4F	SNCL	Compaction table lockword
50	SNLS	IPW$$LH process byte
	SNLP	X'80' - Request for IPW$$LH
	SNHA	X'40' - IPW$$LH is active
51	SNCW	Numner of active workstations
52	SNS1	Status Byte 1
	SNS1STA	X'80' - RJE,SNA SUBTASK ATTACHED
	SNS1VTA	X'40' - VTAM ACTIVE DETECTED BY SUBTASK
53		Reserved
54-	SNAC	VTAM ACB (the ACB is copied from the skeleton ACB as defined in IPW$$I7)

How to Locate: Refer to Figure 152 on page 731 in Chapter 6, "Diagnostic Aids."

SNA Logical Unit Control Block (LUCB)

Definition Macro: IPW$DLU

A logical unit control block (LUCB) is created for each logical unit logged on to the VSE/POWER application.

The LUCB space is allocated together with the SUCB space by the VTAM LOGON exit (IPW$$VE) when the first workstation LU attempts to log on t o VSE/POWER. The number of LUCBs for which storage is reserved depends on the SESSLIM parameter (macro PRMT) of the corresponding remote ID. The LUCB is initiated by the LOGON processors 1 and 2 (IPW$$LH and IPW$$LN).

The LUCB storage is freed together with the SUCB space by IPW$$LF until the last or only LU of a workstation has logged off.

All logical unit control blocks within one workstation are chained together. The pointer to the first LUCB within one workstation is contained in the SUCB, which describes this workstation.

The format of a logical unit control block is as follows.

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(0)	0	STRUCTURE		LUDS	DEFINE DUMMY SECTION
(0)	0	CHARACTER	16	LUSD	SECTION DESCRIPTOR
(10)	16	ADDRESS	4	LUPR	ADDRESS PREVIOUS LUCB
(14)	20	ADDRESS	4	LUNX	ADDRESS NEXT LUCB
(18)	24	ADDRESS	4	LUSU	ADDRESS OF SUCB
(1C)	28	ADDRESS	4	LULWA	ADDRESS OF LUCB LOCKWORD
		..1.		LUBAS	"*-LUDS" LENGTH OF STORAGE DESCRIPTOR PLUS CHAIN POINTERS
(20)	32	CHARACTER	1	LUSL	SELECT INDIC, S = SELECT
(21)	33	CHARACTER	1	LUTT	TERM. TYPE; S = IMMEDIATE
(22)	34	CHARACTER	1	LUTX	TERM. TYPE; SET BY EXITS
(23)	35	BITSTRING	1	LUFS	FREE SESSION INDICATOR
		1...		LUF1	"X'80'" SESSION IS IN USE
(24)	36	ADDRESS	4	LUW1	RDR 1 WORKSPACE ADDRESS
(28)	40	ADDRESS	4	LUW2	RDR 2 WORKSPACE ADDRESS, OBTAINED FROM SUCB
(2C)	44	SIGNED	4	LUCD	VTAM CID
	SESSION ACCOUNT INFORMATION				
(30)	48	CHARACTER	48	LUAR (0)	SESSION ACCOUNT RECORD
(30)	48	CHARACTER	8	LUDY	DATE = C'MM/DD/YY'
(38)	56	CHARACTER	4	LUST	SIGNON TIME = X'0HHMMSSF'
(3C)	60	CHARACTER	4	LUET	SIGNOFF TIME = X'0HHMMSSF'
(40)	64	CHARACTER	16	LUUI	USER INFORMATION
(50)	80	CHARACTER	8	LULU	LOGICAL UNIT NAME
(58)	88	BITSTRING	1	LUCF1	LUCB ACCOUNT FLAG BYTE 1
		1...		LUCE20	"X'80'" LUDY DATE IS 20YY CENTURY
(59)	89	CHARACTER	1		RESERVED

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(5A)	90	CHARACTER	1	LUAI	IDENTIFIER FOR ACCT RECORD
(5B)	91	BITSTRING	1	LUCN	SESSION TERMINATION CODE
		.... ..1.		LUAL	"X'02'" ABNORMAL TERMINATION
		1		LUNL	"X'01'" NORMAL (SIGNOFF OR LOGOFF)
(5C)	92	CHARACTER	4	LURI (0)	REMOTE IDENTIFIER
(5C)	92	BITSTRING	1	LURB	BINARY FORMAT
(5D)	93	CHARACTER	3	LURC	CHARACTER FORMAT
	RESTART INFORMATION				
(60)	96	SIGNED	4	LURS (0)	RESTART INFORMATION
(60)	96	BITSTRING	1	LURX	RESTART FUNCTION INDEX
(61)	97	BITSTRING	3	LURP	RESTART PAGE COUNT
	LIST AND PUNCH CHARACTERISTICS				
(64)	100	ADDRESS	4	LUPH	PTR TO DEVICE IN SUCB
(68)	104	BITSTRING	1	LULO	LIST OUTPUT SUPPORT
		1...		LULAS	"X'80'" ASCII
		.1..		LULCM	"X'40'" COMPRESSION
		..1.		LULTR	"X'20'" TRANSPARENCY
		...1		LULSP	"X'10'" SPANNING
		.... 1...		LULIR	"X'08'" INTER-RECORD SEPARATOR
		.... .1..		LULXL	"X'04'" XLATION OF CHAR BELOW BLANK
		1		LULCP	"X'01'" COMPACTION
(69)	105	BITSTRING	1	LUPO	PUNCH OUTPUT SUPPORT
		1...		LUPAS	"X'80'" ASCII
		.1..		LUPCM	"X'40'" COMPRESSION
		..1.		LUPTR	"X'20'" TRANSPARENCY
		...1		LUPSP	"X'10'" SPANNING
		.... 1...		LUPIR	"X'08'" INTER-RECORD SEPARATOR
		1		LUPCP	"X'01'" COMPACTION
(6A)	106	BITSTRING	1	LUPD	PDIR INFORMATION BYTE
		1...		LUPS	"X'80'" PDIR OUTBOUND ALLOWED
(6B)	107	BITSTRING	1	LUAD	CARD/DOCUMENT FLOW
		1...		LUACI	"X'80'" CARD INBOUND ALLOWED
		.1..		LUACO	"X'40'" CARD OUTBOUND ALLOWED
		..1.		LUAXI	"X'20'" BASIC EX MEDIA IB ALLOWED
		.... 1...		LUALI	"X'08'" DOCUMENT INBOUND ALLOWED
		.... .1..		LUALO	"X'04'" DOCUMENT OUTBOUND ALLOWED
(6C)	108	SIGNED	4	(0)	FULLWORD-BOUNDARY
(6C)	108	CHARACTER	8	LUOC (0)	ACT. COMPACT. TABLE OUT
(6C)	108	CHARACTER	4	LUO1	COMPACTION TABLE NAME
(70)	112	ADDRESS	4	LUO2	POINTER TO COCB ENTRY
	PROCESS CONTROL SECTION				
(74)	116	ADDRESS	4	LUTC (0)	START OF TCBS FOR LU
(74)	116	ADDRESS	4	LURT	RDR|LGN|LGF TCB ADDRESS
(78)	120	ADDRESS	4	LULT	LST|PUN TCB ADDRESS
(7C)	124	ADDRESS	4	LUMT	MSG TCB ADDRESS
(80)	128	ADDRESS	4	LUTI	RDR2 TCB ADDRESS
(84)	132	ADDRESS	4	LUTH	LGH TCB ADDRESS
		.... .1.1		LUTL	"(*-LUTC)/4" NUMBER OF TCBS
(88)	136	BITSTRING	1	LUA1	ACTION BYTE
		1...		LURO	"X'80'" REQUEST LOGON
		.1..		LURR	"X'40'" REQUEST START READER
		..1.		LUSG	"X'20'" REQUEST INTERRUPT LST|PUN ON SIGNAL
		...1		LUMR	"X'10'" REQUEST INTERRUPT LST|PUN FOR OUTBOUND MESSAGE
		.... 1...		LUII	"X'08'" REQUEST INTERRUPT INBOUND FOR INBOUND
		.... .1..		LUSS	"X'04'" REQUEST STOP SESSION
		.... ..1.		LUGO	"X'02'" REQ GO/SETUP CMND,OR NEW FORMS

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
		1		LUCR	"X'01'" REQUEST FOR RESTART CMND
(89)	137	BITSTRING	1	LUP1	PROCESS BYTE
		1...		LUPF	"X'80'" LOGOFF IN PROCESS
		.1..		LUSRD	"X'40'" START READER SWITCH
		..1.		LUPS2	"X'20'" PROCESS SUSPEND2
		...1		LUP1RL	"X'10'" RELEASE LUCB AFTER SEND RPL
(8A)	138	BITSTRING	1	LUS1	STATUS BYTE ONE
		1...		LUBB	"X'80'" BB REJECT INDICATOR
		.1..		LUSO	"X'40'" LOGON COMPLETED
		..1.		LUBBC	"X'20'" CONTENTION FOUND BY IB
		...1		LUS1XX4	"X'10'" RESERVED
		.... 1...		LUS1XX5	"X'08'" RESERVED
		.... .1..		LUS1XX6	"X'04'" RESERVED
		1		LUBBO	"X'01'" 1 - BB REJECT BY $OB 0 - BB REJECT BY $MP
(8B)	139	BITSTRING	1	LUS2	STATUS BYTE 2
		1...		LUBC	"X'80'" CHANGE DIRECTION
		.1..		LUOO	"X'40'" LST/PUN SUSPENDED FOR MSG
		..1.		LUOI	"X'20'" LST/PUN SUSPENDED FOR INBOUND
		...1		LUIS	"X'10'" INBOUND SUSPENDED FOR INBOUND
		.... 1...		LUWL	"X'08'" WAITING FOR LUSTATUS
		.... .1..		LUS2XX6	"X'04'" RESERVED
		.... ..1.		LUS2XX7	"X'02'" RESERVED
		1		LUS2XX8	"X'01'" RESERVED
(8C)	140	BITSTRING	1	LUBR	BRACKET STATE
(8D)	141	BITSTRING	1		UNUSED
(8E)	142	SIGNED	2	LUBS	BUFFER SIZE
(90)	144	SIGNED	2	LUBSL	BUFFER SIZE LOGON PROCESS
(92)	146	SIGNED	2	LUSEQNO	LAST RECEIVED SEQ NUMBER
(94)	148	SIGNED	4	LUSR	SAVE LOSTERM REASON CODE
(98)	152	SIGNED	2	LUBSI	INBOUND RU BUF SIZE
(9A)	154	SIGNED	2	LUBSO	OUTBOUND RU BUF SIZE
(9C)	156	SIGNED	2	LUBSAO	ACT OUTBOUND RU BUF SIZE
ALIGN TO LINE-BOUNDARY AND FILL WITH ZEROS					
(9E)	158	ADDRESS	1	(0)	
		1.1.		LULN	"*-LUDS" LENGTH OF CONTROL BLOCK

SNA Logon Request Control Block (LRCB)

Definition Macro: IPW$DLR

A LOGON request control block contains information for 6 LOGON requests to the VSE/POWER application. All LOGON request control blocks are chained. The pointer to the first LRCB is contained in the SNA control block (SNCB).

Information about LOGON requests are stored in the LRCB by the LOGON exit of the SNA manager. The LOGON processor processes the LOGON requests to build SUCB/LUCBs.

The format of a LOGON request control block is as follows.

Bytes Hex.	Label of Field	Description/Function
00-0F	LRSD	Storage descriptor (LRCB)
10-13	LRNX	Pointer to next LRCB
14-17		Reserved
18	LRLC	Length of one LRCB
19	LRLB	Length of one LRUB
1A	LRAL	No. of total LRUBs in LRCB
1B	LRUS	No. of active LRUBs in LRCB
1C-1F		Reserved
20-7F	LRAU	Space for six LRUBs
20-2F		First LRUB entry
20-23	LRAC	ACB address
24-2B	LRLU	LU-name
2C	LRST	Status (X'01' indicates active entry)
2D-2F	LRLM	Length of LOGON message

SNA Remote Control Block (RMCB)

Definition Macro: IPW$DRM

The SNA remote control block consists of:

- General information that is not required in real storage for RJE,SNA processing.
- A general work space to be used by any SNA routine that cannot obtain virtual storage via the VSE/AF GETVIS macro.
- Translate tables to convert EBCDIC characters to ASCII and vice versa.
- Remote entries for each remote ID specified in the PRMT macro at VSE/POWER generation.

Offset Hex	Type	Len	Name (Dim)	Description
(0)	CHARACTER	16	RMSD	SECTION DESCRIPTOR
(10)	SIGNED	4		RESERVED
(14)	CHARACTER	3		RESERVED
(17)	ADDRESS	1	RMAL	ACB PASSWORD LENGTH
(18)	CHARACTER	8	RMAP	ACB PASSWORD
(20)	CHARACTER	2048	RMGP	GENERAL PURPOSE WORKSPACE This area is serially accessible by SNA tasks that cannot obtain virtual storage via the VSE/Advanced Functions GETVIS macro. Access is regulated by a lockword (SNRL) located in the SNA control block (SNCB).
	THE FOLLOWING TRANSLATE TABLES ARE USED TO CONVERT EBCDIC CHARACTERS TO ASCII AND VICE VERSA.			
(820)	CHARACTER	256	RMEA	EBCDIC TO ASCII
(920)	CHARACTER	128	RMAE	ASCII TO EBCDIC
	FOLLOWING SAVE AREAS ARE FOR THE SNA MANGER			
(9A0)	DBL WORD	8		
(9A0)	CHARACTER	128	RMSNSS (0)	SUBTASK SAVE AREA
(9A0)	CHARACTER	8	RMSNSN	SUBTASK NAME
(9A8)	BITSTRING	120	RMSNSR	REGISTER SAVE AREA
(A20)	CHARACTER	8	RMSNAN	SUBTASK NAME
(A28)	BITSTRING	1	RMSNAS	AB REGISTER SAVE AREA
(B00)	BITSTRING	72	RMSNRA	RECEIVE ANY VTAM SAVE AREA
(B48)	BITSTRING	72	RMSNST	SETLOGON VTAM SAVE AREA
	FOLLOWING ARE THE REMOTE ENTRIES FOR EACH SNA REMID SPECIFIED BY THE PRMT MACRO AT GENERATION TIME.			
(B90)	BITSTRING	1	RMSR	NR OF SNA REMOTES ENTRIES
(B91)	BITSTRING	1	RMFR	FIRST SNA REMOTE ID
(B92)	BITSTRING	1	RMHR	LAST SNA REMOTE ID
(B93)	BITSTRING	1	RMNC	NUMBER 1K BLOCKS - CMPACT
	EXPRESSION		RMLN	"*-RMDS" LENGTH REMOTE CTL BLK LESS ENTRIES
(B94)	CHARACTER	32	RMRM (0)	REMOTE ENTRIES The number of remote entries, which are 32 bytes long, depends on the number of SNA remote units specified in the PRMT macro at VSE/POWER generation.
	The following is a layout of a remote entry			

Offset Hex	Type	Len	Name (Dim)	Description
(B94)	CHARACTER	1	RMPR	PUNCH ROUTING REMOTE ID
(B95)	CHARACTER	1	RMLR	LIST ROUTING REMOTE ID
(B96)	CHARACTER	30	RMRI (0)	REMOTE REFERENCE INFO
(B96)	CHARACTER	2	RMBS	BUFFER SIZE
(B98)	CHARACTER	1	RMTT	TERMINAL TYPE
(B99)	CHARACTER	1	RMTF	TERMINAL FEATURES
	1...		RMCS	"X'80'" ..CONSOLE SPECIFIED
	.1..		RMXL	"X'40'" ..TRANSLATION REQUESTED
(B9A)	CHARACTER	1		RESERVED
(B9B)	CHARACTER	1	RMPL	PASSWORD LENGTH
(B9C)	CHARACTER	8	RMPW	PASSWORD
	EXPRESSION		RMRF	"RMPL" IF REF SPECIFIED IN PRMT MACRO, FIELD = 'FF'X
	EXPRESSION		RMRN	"RMBS+1" IF REF SPECIFIED IN PRMT MACRO, FIELD = REFERENCED REMID
(BA4)	CHARACTER	4	RMCN	COMPACT NAME
(BA8)	ADDRESS	4	RMLU (0)	LU ADDRESS FIELD
(BA8)	BITSTRING	1	RMNL	NUMBER OF LU NAMES
(BA9)	ADDRESS	3	RMLA	ADDRESS OF FIRST LU NAME
(BAC)	SIGNED	2	RMSL	SESSION LIMIT
(BAE)	SIGNED	2	RMML	MAX RECORD SIZE FOR LIST
(BB0)	BITSTRING	4		RESERVED
(BB4)	EXPRESSION		RMNX	"*" END OF REMOTE ENTRY

END OF RMCB IS FORCED TO NEXT PAGE BOUNDARY BY INITIALIZATION

How to Locate: Refer to Figure 152 on page 731 in Chapter 6, “Diagnostic Aids.”

SNA Session Request Queue (SRQE)

Definition Macro: IPW$DRQ

Bytes Hex.	Label of Field	Description/Function
000-00F	SRQESD	Section descriptor
010-011	SRQETLGF	SRQE length (multiple 128)
012-013		Reserved
014-017	SRQENPTR	Next in chain pointer
018-01B	SRQETPTR	Task pointer belonging to
01C-01F	SRQEANCB	Address of node control block
020-021	SRQEALEN	Actual length of BIND-RU

- Status:

022	SRQESTA	SRQE status byte
023	SRQERC	Reason code used in NSEXIT
024-025	SRQESSMO	Sense modifier

- This part will contain the BIND-RU.

026-049	SRQEBDRU	Reserved for BIND-RU
026-03F	SRQEBIND	BIND-RU area
040	SRQEPLUL	PLU-name length
041-048	SRQEPLU	PLU-name
049		

- This part contains the VTAM emergency save area.

04A-04B		Unused
04C-093	SRQESAVE	Reserved for VTAM save area
094-097	SRQESR13	Save reg. 13, if VTAM-macro
098-09B		Reserved

- This part contains the RPL after a BIND-RU has been received.

09C-0FF	SRQERPL	RPL area

- This part contains the NIB after a BIND-RU has been received.

100-13F	SRQENIB	NIB area

SNA Unit Control Block (SUCB)

Definition Macro: IPW$DSU

An SNA unit control block is created for each workstation that is logged on to the VSE/POWER application with one logical unit. All SNA unit control blocks are chained together.

The SUCB is allocated from the VTAM LOGON exit (IPW$$VE) when the first workstation LU attempts to log on to VSE/POWER. It is initialized by the LOGON processor 1 (IPW$$LH). The SUCB storage is freed by IPW$$LF until the last or only LU of a workstation has logged off.

Bytes	Label	
Hex.	of Field	Description/Function
00-0F	SUSD	Storage descriptor (SUCB)
10-13	SUNX	Address of next SUCB

- General Accounting Information

14-1F	SUAR	General Information
14-1B	SUDY	Date = C'MM/DD/YY'
10-1F	SURI	Remote Identifier
1C	SURB	- Binary format
1D-1F	SURC	- Character format

- List, Punch and Reader Device Characteristics

20	SULR	List routing remid
21	SUPR	Punch routing remid
		Device status values for the following devices:
	SUHS	X'80' - Device started
	SUHU	X'40' - Device available
	SUHO	X'20' - Output available
	SUSKIP	X'10' - Skip to channel 1 requested
22-23		Reserved
24-27	SUL1P	Printer 1 - C'LST1'
28	SUL1S	Device status
29-2B	SUL1L	Pointer to LUCB
2C-2F	SUL1F	Forms ID
30-33	SUL1C	List output classes
34-37	SUL2P	Printer 2 - C'LST2'
38	SUL2S	Device status
39-3B	SUL2L	Pointer to LUCB
30-3F	SUL2F	Forms ID
40-43	SUL2C	List output classes
44-47	SUL3P	Printer 3 - C'LST3'
48	SUL3S	Device status
49-4B	SUL3L	Pointer to LUCB
40-4F	SUL3F	Forms ID
50-53	SUL3C	List output classes
54-57	SUP1P	Punch - C'PUN1'
58	SUP1S	Device status
59-5B	SUP1L	Pointer to LUCB
5C-5F	SUP1F	Forms ID
60-63	SUP1C	Punch output classes
64-67	SUR1P	Reader - C'RDR1'

Bytes Hex.	Label of Field	Description/Function
68	SUR1S	Device status
69-6B	SUR1L	Pointer to LUCB
6C-6F	SUR1F	Forms ID (not used by reader)
70-73	SUR1C	Reader class - C'A' (initialized by IPW$$IB)
74-77	SUX1P	Exchange media reader - 'RDR2'
78	SUX1S	Device status
79-7B	SUX1L	Pointer to LUCB
7C-7F	SUX1F	Forms ID (not used by reader)
80-83	SUX1C	Reader class - C'A' (initialized by IPW$$IB)
84-87	SUC1P	Console - C'CON1'
88	SUC1S	Device status
89-8B	SUC1L	Pointer to LUCB
8C-8F	SUC1C	Forms ID (not used by console)
90-93	SUC1C	Console class - C'A' (initialized by IPW$$IB)
94	SUHD	Device List delimiter
95-96		Reserved
97	SUDLS	Device select indicator

- Compaction Table Information for Outbound (referred to by SOOC)

98-9B	SUO1	Name of default table
9C-9F	SUO2	Address of default table virtual
A0	SUAD	Card/document flow X'80' - Card inbound allowed X'40' - Card outbound allowed X'08' - Document inbound allowed X'04' - Document outbound allowed

- Message Control Section

A1	SUMR	Message request status X'80' - Message processor for work station is active X'40' - Request to interrupt IPW$$OB for outbound message was issued
A2-A4	SUMRL	Pointer to the LUCB with the suspending IPW$$OB
A5		Unused
A6-A7	SUMN	No. of messages
A8	SUMC	Subchain index
A9	SUMD	Temporary delete chain index
AA	SUTY	Terminal type
AB	SUTF	Terminal features X'80' - Console specified

- Miscellaneous

AC-AF	SUWLW	Address of workstation lockword
B0-B3	SULKA	Address of Lockword Table
B4-B7	SUW1	Inbound work space address
B8		Reserved
B9-BB	SUWSL	Pointer to LUCB. If set then work space is in use by the LUCB being pointed to.
BC-BF	SUPL	Pointer to first LUCB
C0-C1	SUN1	No. of attached LUCBs
C2-C3	SUN2	No. of active LUCBs
C4-C5	SURUSZ	MAX RU BUF SIZE FROM PRMT

How to Locate: Refer to Figure 152 on page 731 in Chapter 6, “Diagnostic Aids.”

SNA Work Area (WACB)

Definition Macro: IPW$DWA

This work space is reserved for and used by each logical unit processing routine (RDR, LST, PUN, and MSG).

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(0)	0	STRUCTURE		WADS	, DEFINE DUMMY SECTION WADA MOVED FROM BEHIND WASV
(0)	0	CHARACTER	256	WADA	PLS DYNAMIC AREA
(100)	256	CHARACTER	16	WASD	SECTION DESCRIPTOR
(110)	272	CHARACTER	16	WABC (0)	BUFFER CONTROL FIELDS
(110)	272	SIGNED	4	WARC	RESIDUAL COUNT IN BUFFER
(114)	276	ADDRESS	4	WACR	CURRENT POSISTION IN BUFFER
(118)	280	ADDRESS	4	WABI	ADDRESS BUFFER TO SEND/RECEIVE
(11C)	284	ADDRESS	4	WABP	ADDRESS BUFFER IN PROCESS (FILL)
(120)	288	SIGNED	2	WARL	LOGICAL RECORD LENGTH
(122)	290	SIGNED	2		RESERVED FOR MSG SERVICE AND COMPRESSION
(124)	292	CHARACTER	1	WALR (136)	LOGICAL RECORD
(1AC)	428	ADDRESS	4	WALR2A	LOGICAL RECORD 2 ADDRESS
(1B0)	432	SIGNED	2	WALR2L	LOGICAL RECORD 2 LENGTH
(1B2)	434	BITSTRING	2		RESERVED
	16 BYTES RESERVED FOR RU SIZE EXCEEDING 256 BYTES				
(1B4)	436	ADDRESS	4	WARU1P	RU 1 POINTER
(1B8)	440	ADDRESS	4	WARU2P	RU 2 POINTER
(1BC)	444	SIGNED	2	WARUBL	RU BUFFER LENGTH $RSV
(1BE)	446	SIGNED	2	WARUSZ	RU SIZE
(1C0)	448	ADDRESS	4		RESERVED
	PROCESSING SWITCH AND STATUS BYTES				
(1C4)	452	BITSTRING	1	WASW	PROCESSING SWITCHES BITS 0-5 REDEFINED
		.... ..1.		WACE	"X'02'" END OF FILE
		1		WALI	"X'01'" LOGICAL INTERFACE OPEN
(1C5)	453	BITSTRING	1	WAST	STATUS BYTE
(1C6)	454	BITSTRING	1	WASS	DATA STREAM STATE
(1C7)	455	BITSTRING	1	WACS	CHAIN STATE
(1C8)	456	BITSTRING	1	WAPR	PROCESS OPTIONS IN EFFECT
		1...		WAAS	"X'80'" ASCII SUPPORT
		.1..		WACM	"X'40'" COMPRESSION SUPPORT
		..1.		WATR	"X'20'" TRN SUPPORT
		...1		WASP	"X'10'" SPANNING SUPPORT
		.... 1...		WARS	"X'08'" IRS SUPPORT
		.... .1..		WAXL	"X'04'" TRANSLATION REQUESTED
		1		WACP	"X'01'" COMPACTION SUPPORT
(1C9)	457	BITSTRING	1	WACI	COMPACTION INDICATOR
		1...		WAUI	"X'80'" IF ON, USE COUNT INCREASED
		.1..		WACF	"X'40'" COMPACTION TABLE FOUND
(1CA)	458	BITSTRING	2		RESERVED
(1CC)	460	SIGNED	4	WAPH	SAVE AREA FOR LUPH FOR INTERRUPTING PROCESSORS
(1D0)	464	SIGNED	4	WASN	ERROR SENSE BYTES
(1D4)	468	ADDRESS	2	WAMN	ERROR MESSAGE NUMBER
(1D6)	470	CHARACTER	2		RESERVED
(1D8)	472	SIGNED	4	WAER	ERROR ROUTINE ADDRESS
(1DC)	476	CHARACTER	112	WARP	RPL + 12 BYTES FOR FUTURE EXPANSION

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(24C)	588	CHARACTER	72	WASV	REGISTER SAVE AREA
(294)	660	CHARACTER	12		RESERVED
(2A0)	672	CHARACTER	8	WAFM	FUNCTION MANAGEMENT HEADER +2 BYTES
(2A0)	672			WALN	"*-WADS" LENGTH OF WADS WITHOUT BUFFERS
(2A0)	672			WABF	"*" TWO BUFFERS
CONSTANTS					
(2A0)	672	SIGNED		WABL	"256" NORMAL SNA BUFFER SIZE
(2A0)	672	SIGNED		WABM	"512" MAXIMUM SNA BUFFER SIZE
		1...		WABS	"B'10000000'" BETWEEN STATE SETTING
		.1..		WAIN	"B'01000000'" IN STATE SETTING
		..1.		WAEP	"B'00100000'" END STATE PENDING SETTING
		...1		WASA	"B'00010000'" ABORT STATE SETTING
		.... 1...		WASH	"B'00001000'" SUSPEND STATE SETTING
PROCESSING SWITCHES AS USED BY PROCESSORS					
(2A0)	672			WAIS	"WASW" PROCESSING SWITCHES AS USED BY IB
		1...		WAIC	"X'80'" PROCESSING CONSOLE
		.1..		WAIR	"X'40'" PROCESSING READER
		..1.		WAUB	"X'20'" UNCONDITIONAL END BRACKET
		...1		WATI	"X'10'" IMM. TERMINATION REQUIRED
		.... 1...		WASR	"X'08'" RESUME REQUESTED X'02' END OF FILE X'01' LOGIC. INTERFACE OPEN
(2A0)	672			WAOS	"WASW" PROCESSING SWITCHES AS USED BY OB
		1...		WAOF	"X'80'" EOF REACHED
		.1..		WAOJ	"X'40'" EOJ OR CHAIN REACHED
		..1.		WAOL	"X'20'" END OF LOGICAL RECORD REACHED
		...1		WAOR	"X'10'" END OF RU REACHED
		.... 1...		WAOU	"X'08'" SETUP/GO IN PROCESS
		.... .1..		WAF3	"X'04'" FMH3 SEND INDICATOR X'02' END OF FILE X'01' LOGIC. INTERFACE OPEN
(2A0)	672			WAMS	"WASW" PROCESSING SWITCHES AS USED BY MP
		..1.		WAMC	"X'20'" COMPONENT NOT AVAILABLE
		...1		WAMR	"X'10'" END OF RU REACHED
RECORD AREAS AS USED BY PROCESSORS					
(124)	292	BITSTRING	7	WABD	LOGON PROCESSOR FOR BIND IMAGE
(2A8)	680			WANB	"*" NIB AND BIND-AREA
(2A0)	672	CHARACTER	6	FMH (0)	FUNCTION MANAGEMENT HEADER IN DWA
(2A0)	672	BITSTRING	1	FMHLN	FMH LENGTH BYTE
(2A1)	673	BITSTRING	1	FMHTYP	FMH TYPE BYTE
		1...		FMHC	"X'80'" CONCATENATION
		..11 1111		FMHTYPE	"B'00111111'" FMH TYPE 1
(2A2)	674	BITSTRING	1	FMHSEL	FMH SELECT BYTE
		1...		FMHDS	"B'10000000'" DEMAND SELECT
		.111		FMHMEDIA	"B'01110000'" DEVICE SELECT
		....		FMHCNS	"B'00000000'" CONSOLE
		..1.		FMHCRD	"B'00100000'" CARD READER
		..11		FMHPRT	"B'00110000'" PRINTER
		 1111		FMHLOGAD	"B'00001111'" LOGICAL ADDRESS
(2A3)	675	BITSTRING	1	FMHFLAG	FMH FLAG BYTE
		1...		FMHSTACK	"X'80'" ADS SEND BY RECEIVER
(2A4)	676	BITSTRING	1	FMHPROP	FMH PROPERTIES BYTE
		111.		FMHDSS	"B'11100000'" DATA STREAM STATE
		....		FMHRDS	"B'00000000'" DATA STREAM RESUME
		..1.		FMHEDS	"B'00100000'" DATA STREAM END
		.1..		FMHBDS	"B'01000000'" DATA STREAM BEGIN
		.11.		FMHBEDS	"B'01100000'" DATA STREAM BEGIN AND END
		1...		FMHIDS	"B'10000000'" DATA STREAM INTERRUPT(SUSPEND=SDS)
		1.1.		FMHADS	"B'10100000'" DATA STREAM ABORT
		...1 1111		FMHDSC	"B'00011111'" DATA STREAM CHARACTERISTICS

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
		...1		FMHDST	"B'00010000'" BASIC EXCHANGE
		.... .1..		FMHCMI	"B'00000100'" COMPRESSION
		.... ..1.		FMHCPI	"B'00000010'" COMPACTION
(2A5)	677	BITSTRING	1	FMHERCL	FMH BASIC EXCHANGE LENGTH BYTE

How to Locate: Refer to Figure 152 on page 731 in Chapter 6, "Diagnostic Aids."

Source Library Member Element (SLME)

Definition Macro: IPW$DSL SLME=YES

A source statement library member element is reserved by IPW$$SL for each book. The SMLEs are chained together whereby the top SLME represents the deepest nesting level. The SLMEs are anchored to the SLWA.

Offset Hex	Type	Len	Name (Dim)	Description
(0)	ADDRESS	4	SLMNXT	NEXT SLME IN CHAIN
(4)	BITSTRING	1	SLMFL1	FLAG BYTE 1
	1...		SLMPRQ	"X'80'" ..POINT REQUIRED
	.1..		SLMEOM	"X'40'" ..END OF MEMBER
(5)	BITSTRING	1	SLMFL2	FLAG BYTE 2
	1...		SLICCF	"X'80'" ..SLME IN ICCF FORMAT
	.1..		SLVSESL	"X'40'" ..SLME FOR PRIVATE LIB CHAIN
	..1.		SLDISLIB	"X'20'" ..DON'T DISCONNECT LIBR
	...1		SLDISSEC	"X'10'" ..DO DISCONNECT SECURITY
(6)	BITSTRING	1	SLRTRC	FIND RETRY COUNTER (ICCF)
	.... 1.1.		SLRTMX	"10" ..MAXIMUM NUMBER OF RETRIES
(7)	BITSTRING	1		RESERVED
(8)	SIGNED	2	SLGP	GEN. PURPOSE BYTE SAVE AREA
(A)	BITSTRING	2		RESERVED
	.... 11..		SLMARG	"*" LIBR ARG / ICCF SUBBLOCK
	LAYOUT OF LIBRARIAN MEMBER ARGUMENT LIST			
(C)	CHARACTER	8		RESERVED
(14)	CHARACTER	8	LARGMTYP	MEMBER TYPE
(1C)	CHARACTER	8	LARGNAM	MEMBER NAME
(24)	CHARACTER	12	SLMNOPO	NOTE/POINT ARGUMENT FIELD
	LAYOUT OF ARGUMENT SUBBLOCK FOR ICCF			
(C)	CHARACTER	8	SLMNAME	MEMBER NAME
(14)	CHARACTER	8	SLMPWRD	MEMBER PASSWORD
(1C)	CHARACTER	8	SLMUSID	USERID OF ORIGINATOR
(24)	SIGNED	2	SLMLIBN (3)	BINARY SUBLIB NUMBERS
(2A)	BITSTRING	18	SLMICARG	ICCF THIRD WORKSPACE
	..11 11..		SLMELN	"*-SLMEDS" LENGTH OF CONTROL BLOCK

Source Library Work Area (SLWA)

Definition Macro: IPW$DSL SLWA=YES

This work space is reserved and used by phase IPW$$SL and provides storage to read records from a source statement library. The work space is anchored to the partition control block of the partition concerned.

Offset Hex	Type	Len	Name (Dim)	Description
(0)	CHARACTER	16	SLWASD	STORAGE DESCRIPTOR
	COMMUNICATION SWITCHES			
(10)	CHARACTER	1	SLRS	READ SSL SWITCH
				C'R' ..READ REQUEST
(11)	CHARACTER	1	SLRR	READ RDR SWITCH
				C'R' ..READ REQUEST
				C'I' ..IGNORE RDR RECORD
(12)	BITSTRING	1	SLF1	FLAG BYTE 1
	1...		SLEOM	"X'80'" ..END OF MEMBER
	.1..		SLFPVT	"X'40'" ..PRIVATE LIBDEF CHAIN
	.... 1...		SLNCNT	"X'08'" ..IGNORE CONTINUATION CARD (NOT $$SLI)
(13)	CHARACTER	1		RESERVED FOR FUTURE USE
	BUFFER CONTROL INFORMATION			
(14)	ADDRESS	4	SLCREC	CURRENT RECORD ADDRESS
(18)	ADDRESS	4	SLLREC	ADDR OF LAST REC IN BUFFER
	.1.1		SLRLEN	"80" ..SSL RECORD LENGTH
(1C)	CHARACTER	80	SLRBUF (10)	BUFFER AREA
	EXPRESSION		SLRBLN	"*-SLRBUF" ..BUFFER AREA LENGTH
	POINTERS AND SAVE AREAS			
(33C)	ADDRESS	4	SLAPDB	ADDR OR PART CNTRL BLK
(340)	ADDRESS	4	SLASRB	ADDR OF SERVICE REQUEST BLK
(344)	ADDRESS	4	SLSLME	ADDR OF CURR SL-MEMBER ELEM
(348)	ADDRESS	4	SLIBUF	ADDR OF ICCF PROCESS BUFFER
(34C)	ADDRESS	4	SLSAVA (14)	SAVE AREA USED BY ASYN SERV
(384)	ADDRESS	4	SLDALN	DATA NAME LENGTH SAVE AREA
(388)	ADDRESS	4	SLDAPL	VSE SECURITY APL AREA PNTR
(38C)	ADDRESS	4	SLLBAER	ADDR OF BAD LIB.SUBLIB NAME
	NON-VSE/POWER LIBDEF CHAIN: LIBRARY AND SUBLIBRARY SEARCH NAMES			
(390)	CHARACTER	7	SLMLIB1	LIBRARY 1 NAME
(397)	CHARACTER	8	SLMSUB1	SUBLIBRARY 1 NAME
(39F)	CHARACTER	7	SLMLIB2	LIBRARY 2 NAME
(3A6)	CHARACTER	8	SLMSUB2	SUBLIBRARY 2 NAME
(3AE)	CHARACTER	7	SLMLIB3	LIBRARY 3 NAME
(3B5)	CHARACTER	8	SLMSUB3	SUBLIBRARY 3 NAME
	..1. 11.1		SLSERLN	"*-SLMLIB1" LENGTH OF LIST
(3BD)	CHARACTER	3		UNUSED
(3C0)	CHARACTER	8	SLMSCNM	LIBDEF CHAIN NAME
(3C8)	SIGNED	4		ALIGNMENT

Offset Hex	Type	Len	Name (Dim)	Description
(3C8)	BITSTRING		SLMSCVEC (0)	LIBINFO VECTORS
(3C8)	ADDRESS	4		LIBINFO VECTORS
AF LIBRARIAN ACCESS CONTROL BLOCK USED FOR PRIVATE SUBLIBRARY CHAIN				
(3D8)	SIGNED	4	SLWLACB (0)	
(3D8)	SIGNED	4		LIBINFO
(3DC)	SIGNED	4		
(3E0)	SIGNED	4		
(3E4)	ADDRESS	4		LOCK TABLE ENTRY
(3E8)	SIGNED	4		LAMB
(3EC)	BITSTRING	1		MAIN FUNCTION
(3ED)	BITSTRING	1		LIBTYPE
(3EE)	BITSTRING	1		LIBUSE
(3EF)	ADDRESS	1		CHAIN
(3F0)	BITSTRING	1		
(3F1)	ADDRESS	1		DEFAULT
(3F2)	BITSTRING	2		LOGICAL UNIT
(3F4)	BITSTRING	4		START ADDRESS OF EDT
(3F8)	BITSTRING	4		FOR INTERNAL USE ONLY
(3FC)	BITSTRING	4		FOR INTERNAL USE ONLY
(400)	ADDRESS	1		DEFINE
(401)	ADDRESS	1		LEVEL
(402)	ADDRESS	1		REPLACE=NO
(403)	BITSTRING	1		LBR API OPTION
(404)	BITSTRING	4		MOFIFICATION LEVEL
(408)	BITSTRING	36		LIBRARY DEFINITION TABLE ENTRY
(42C)	BITSTRING	44		SDT ENTRY
(458)	ADDRESS	4		
(45C)	BITSTRING	4		
(460)	BITSTRING	2		PID
(462)	BITSTRING	18		FOR FURTHER USE
(474)	SIGNED	2	LBRD0009	LENGTH OF DTL
(476)	ADDRESS	1		TYPE OF CONTROL
(477)	ADDRESS	1		JC AND VSAM FLAGS
(478)	CHARACTER	12		RESOURCE NAME
(484)	CHARACTER	6		VOLUME ID
(48A)	BITSTRING	1		ECB OF ERQUESTING TASK
(48B)	BITSTRING	1		RETURN CODE OF THE REQUEST
(48C)	BITSTRING	1		FLAG TO POST THE TASK
(48D)	BITSTRING	1		BYTE 3 OF ECB
(48E)	BITSTRING	1		RESERVED
(48F)	BITSTRING	1		RESERVED
(490)	BITSTRING	1		RESERVED
(491)	BITSTRING	1		RESERVED
RPL AND BUFFER USED BY LIBRARIAN				
(494)	SIGNED	4		
(494)	ADDRESS	1		RPL ID FIELD
(495)	BITSTRING	1		. RPL SUBTYPE FIELD
(496)	ADDRESS	2		RPL LENGTH
(498)	BITSTRING	4		. RBA
(49C)	ADDRESS	4		. SEARCH ARGUMENT PTR
(4A0)	ADDRESS	4		. USER I/O AREA
(4A4)	ADDRESS	4		. RECORD LENGTH
(4A8)	ADDRESS	4		. I/O AREA LENGTH
(4AC)	ADDRESS	4		. ACB POINTER
(4B0)	BITSTRING	1		. STRING ID
(4B1)	BITSTRING	1		REQUEST TYPE
(4B2)	ADDRESS	2		. KEY LENGTH
(4B4)	BITSTRING	1		OPTCD BYTE 1
(4B5)	BITSTRING	1		OPTCD BYTE 2
(4B6)	ADDRESS	1		. RESERVED

Offset Hex	Type	Len	Name (Dim)	Description
(4B7)	ADDRESS	1		. TEST AND SET BYTE
(4B8)	BITSTRING	1		. FLAG BYTE
(4B9)	BITSTRING	3		FEEDBACK CODES
(4BC)	ADDRESS	4		. POINTER TO NEXT RPL
(4C0)	BITSTRING	1		. AIX FLAG BYTE
(4C1)	ADDRESS	1		. RESERVED
(4C2)	BITSTRING	2		NUMBER OF POINTERS
(4C4)	ADDRESS	1		. TRANSACTION ID
(4C5)	ADDRESS	3		. RESERVED
(4C8)	BITSTRING	128	SLRPL	REQUEST PARAMETER LIST AREA
(548)	CHARACTER	1540	SLBUF	PROCESS BUFFER AREA
	EXPRESSION		SLPBSZ	"*-SLBUF" LENGTH OF PROCESS BUFFER
(B4C)	BITSTRING	1	SLLBMSGL	LIBRARIAN MESSAGE BUFFER LENGTH
(B4D)	CHARACTER	121	SLLBMSG	LIBRARIAN MESSAGE BUFFER
(C26)	CHARACTER	1		SUPERFLUOS MESSAGE BYTE
(C27)	EXPRESSION		SLWALN	"*-SLDS" LENGTH OF SLWA

Spool Parameter List (SPL)

Definition Macro: SPL TYPE=MAP

The SPL is the means of cross-partition communication between VSE/POWER and another program using the PUTSPOOL, GETSPOOL, or CTLSPOOL interface. When VSE/POWER receives control, the SPL address is located at the user's XECB+5, and spool management initializes the address in the TCB (TCPL) for use by VSE/POWER. The external interface is described in the *VSE/Advanced Functions Macro Reference*.

Bytes Hex.	Label of Field	Description/Function
00	SPLB	Length of spool parameter list
01-03	SPHD	SPL header ('SPL')
04-0B	SPJB	Unique VSE/POWER job name
0C-0D	SPJN	VSE/POWER job number
0E-15	SPPW	Password
16-1D	SPUS	Userid of issuer
1E	SPER	Error feedback
	SPIA	X'80' - Invalid address
	SPLA	X'88' - Invalid SPL address
	SPPA	X'84' - Invalid POWER buffer address
	SPBA	X'82' - Invalid data buffer chain
	SPPP	X'40' - Diagnostic logged by VSE/POWER
	SPFP	X'49' - Task is waiting for queue/account file space. This value will not appear when the PUTSPOOL/GETSPOOL user receives control back from VSE/POWER. The feedback is changed before return to X'09'. The feedback X'49' can only be tested for by users running asynchronously.
	SPSP	X'48' - During PUTSPOOL processing
	SPLP	X'44' - During GETSPOOL processing
	SPCP	X'42' - During CTLSPOOL processing
	SPAP	X'41' - VSE/POWER terminated
	SPUE	X'20' - Processing error
	SPLE	X'28' - Invalid CTLSPOOL request
	SPBE	X'24' - Loop in PUTSPOOL buffer chain; or, more than 4096 buffers used per request
	SPPE	X'22' - GETSPOOL was unable to locate output file by specified job name, job class, and dispatchable VSE/POWER disposition; or, requested output file is in use
	SPSE	X'21' - Buffer area too small (88-byte minimum)
	SPPI	X'10' - Invalid parameter
	SPJI	X'18' - Invalid job name
	SPPWI	X'17' - Invalid password
	SPQI	X'16' - Invalid queue id
	SPCI	X'14' - Invalid class
	SPDI	X'12' - Invalid disposition
	SPOI	X'11' - Invalid command
	SPNR	X'00' - Normal return
	SPLR	X'08' - End-of-data on GETSPOOL
	SPFR	X'09' - **Warning: Task had to wait for queue/ account file space.**

Bytes Hex.	Label of Field	Description/Function
1F	SPER2	Error-feedback byte 2
	SPAI	X'80' - Access inhibited (wrong password)
	SPME	X'01' - Multiple queue entries found
20	SPR1	PUTSPOOL request type
	SPEJ	X'40' - The last data record for internal reader job is contained in this PUTSPOOL request
21	SPR2	CTLSPOOL request type
	SPRP	X'01' - Route to new priority
	SPRD	X'02' - Route to new disposition
	SPRC	X'04' - Route to new class
	SPRJ	X'08' - Route to new remote ID
	SPCX	X'10' - Cancel from RDR queue
	SPSC	X'20' - Scratch from LST queue
	SPST	X'40' - Display job status
	SPPC	X'80' - User-supplied POWER command
22	SPR3	GETSPOOL request type
	SPLD	X'01' - GETSPOOL request
	SPPO	X'02' - Position on Q-record
	SPBR	X'04' - Position on line number
	SPCO	X'08' - Return control characters
23	SPBG	X'10' - Buffered GETSPOOL
	SPR4	CTLSPOOL request-byte 2
	SPOO	X'80' - Spool queue display
	SPQR	X'20' - Queue lookup request
24-2B	SPXR	PUTSPOOL user's XECB name
2C-33	SPXL	GETSPOOL/CTLSPOOL user's XECB name
34-37	SPCB	Address current PUTSPOOL buffer area
38-3B	SPPB	Address user-supplied buffer area for VSE/POWER
	SPMO	X'1C' - Message displacement in buffer from byte 0
3C-3F	SPBL	Data buffer area length
40-43	SPRL	Data record length
	SPRS	Browse control
44	SPSN	Signed browse start control
45-47	SPCT	Browse start line number
48	SPCL	LST output class
49	SPDP	LST output disposition
4A	SPCC	Print/POWER control character
4B	SPSQ	Display job status return
		C'N' - Not on VSE/POWER queues
		C'R' - On RDR queue
		C'L' - On LST queue
		C'P' - On PUN queue
		C'X' - On XMT queue
4C	SPQD	Job disposition on RDR/LST queue
4D-4E		Unused
4F	SPNV	CTLSPOOL new value PRI= DISP= CLASS= REMOTE=
50-53	SPLC	Number of lines/cards
54-57		Reserved for future use

Spool Access Support Parameter List (PWRSPL)

Definition Macro: PWRSPL TYPE=MAP

This macro is used to produce a DSECT for the Spool Access Support (SAS). The format is as follows:

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
					Spool Access Support Parameter List (PWRSPL)
(0)	0	SIGNED	4	SPLDS (0)	START OF SPOOL PARAMETER LIST
(0)	0	CHARACTER	3	SPLGHD	STORAGE DESCRIPTOR SPL
(3)	3	ADDRESS	1	SPLGVM	VERSION AND MOD LEVEL
		...1		SPLGVM1	"X'10'" .. VERSION AND MODIFICATION LVL 10
		..1.		SPLGVM2	"X'20'" .. VERSION AND MOD. LVL 20
		..11		SPLGVM3	"X'30'" .. VERSION AND MOD. LVL 30
(4)	4	CHARACTER	8	SPLGJB	JOB NAME
(C)	12	ADDRESS	2	SPLGJN	JOB NUMBER
(E)	14	BITSTRING	1	SPLGJS	JOB SUFFIX NUMBER
		1...		SPLGJSLA	"X'80'" .. LAST SEGMENT INDICATOR (NOTE: BITS 1 - 7 ARE THE JOB SUFFIX NUMBER(1 - 127) IF ANY)
(F)	15	CHARACTER	1	SPLGCL	JOB CLASS
(10)	16	CHARACTER	8	SPLGPW	PASSWORD
(18)	24	CHARACTER	8	SPLGUS	USER ID OF REQUESTOR
(20)	32	CHARACTER	1	SPLGQI	QUEUE IDENTIFIER
		CHARACTER		SPLGQIR	"C'R'" .. RDR QUEUE IDENTIFIER
		CHARACTER		SPLGQIL	"C'L' .. LST QUEUE IDENTIFIER
		CHARACTER		SPLGQIP	"C'P'" .. PUN QUEUE IDENTIFIER
		CHARACTER		SPLGQIX	"C'X'" .. XMT QUEUE IDENTIFIER
(21)	33	BITSTRING	1	SPLGFLG	FLAG BYTE
	THE FOLLOWING FIELDS DEFINE THE REQUEST TYPES. CONTENTS OF SUBREQUEST BYTE AND FUNCTION BYTES DEPEND ON THE REQUEST TYPE.				
(22)	34	BITSTRING	1	SPLGRQB	REQUEST BYTE
		1		SPLGRPUT	"X'01'" .. PUT REQUEST
		.... ..1.		SPLGRGET	"X'02'" .. GET REQUEST
		11		SPLGRCTL	"X'03'" .. CTL REQUEST
		.... .1..		SPLGRGCM	"X'04'" .. GCM REQUEST
(23)	35	BITSTRING	1	SPLGSRB	SUBREQUEST BYTE
		1		SPLGSRDY	"X'01'" .. DISPLAY JOB / OUTPUT QUEUE ENTRY
		.... ..1.		SPLGSRCN	"X'02'" .. CANCEL JOB
		11		SPLGSRRL	"X'03'" .. RELEASE JOB / OUTPUT QUEUE ENTRY
		.... .1..		SPLGSRHD	"X'04'" .. HOLD JOB / OUTPUT QUEUE ENTRY
		.... .1.1		SPLGSRDL	"X'05'" .. DELETE JOB / OUTPUT QUEUE ENTRY
		.... .11.		SPLGSRAL	"X'06'" .. ALTER JOB / OUTPUT QUEUE ENTRY
		.... .111		SPLGSRCM	"X'07'" .. VSE/POWER COMMAND
		.... 1...		SPLGSRDC	"X'08'" .. DELETE CHECKPOINT INFO
(24)	36	BITSTRING	1	SPLGFB1	FUNCTION BYTE 1
		1		SPLGF1AP	"X'01'" .. APPEND OF INCOMPLETE QUEUE ENTRY
		.... ..1.		SPLGF1RS	"X'02'" .. RESTART OF QUEUE ENTRY
		11		SPLGF1BR	"X'03'" .. BROWSING OF QUEUE ENTRY
		.... .1..		SPLGF1GG	"X'04'" .. GENERIC GET REQUEST
		.... .1.1		SPLGF1QM	"X'05'" .. PUT:QUEUE COMPLETION MSG

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
		.... .11.		SPLGF1KM	"X'06'" .. GCM:RETR. AND KEEP MSG
		.... .111		SPLGF1DM	"X'07'" .. GCM:RETR. AND DELETE MSG
		.... 1...		SPLGF1RM	"X'08'" .. GCM:REMOVE FLAGGED MSG
		.... 1..1		SPLGF1QQ	"X'09'" .. PUT:QUEUE JOB EVENT MSG
		.... 1.1.		SPLGF1PM	"X'0A'" .. GCM:PURGE MESSAGE QUEUE
(25)	37	BITSTRING	1	SPLGFB2	FUNCTION BYTE 2
		1		SPLGF2CL	"X'01'" .. ALTER CLASS
		.... ..1.		SPLGF2DP	"X'02'" .. ALTER DISPOSITION
		11		SPLGF2CP	"X'03'" .. ALTER COPY NUMBER
		.... .1..		SPLGF2CM	"X'04'" .. ALTER COMPACTION TABLE NAME
		.... .1.1		SPLGF2RE	"X'05'" .. ALTER REMOTE ID
		.... .11.		SPLGF2PR	"X'06'" .. ALTER PRIORITY
		.... .111		SPLGF2SY	"X'07'" .. ALTER SYSTEM IDENTIFIER
		.... 1...		SPLGF2TN	"X'08'" .. ALTER DESTINATION NODE NAME
		.... 1..1		SPLGF2TU	"X'09'" .. ALTER DESTINATION USER ID
		.... 1.1.		SPLGF2MR	"X'0A'" .. RELEASE GETS COMPL. MSG
(26)	38	CHARACTER	8	SPLGNV	FIELD CONTAINING NEW VALUE FOR ALTER OR EXTRA CLASSES
(26)	38	CHARACTER	3	SPLGACLS	EXTRA CLASSES FOR GENERIC GET
(2E)	46	ADDRESS	1	SPLGOPT	OPTION BYTE 1
		1...		SPLGOSEP	"X'80'" .. RETURN SEPARATOR PAGES/CARDS
		.1..		SPLGOFCC	"X'40'" .. FEED BACK IMMEDIATE COMMANDS
		..1.		SPLGOALL	"X'20'" .. PASS ALL COPIES OF QUEUE ENTRY
		...1		SPLGOFIX	"X'10'" .. RETURN FIXED FORMAT QUEUE DISPLAY
		.... 1...		SPLGONOW	"X'08'" .. NOWAIT OPTION
		.... .1..		SPLGOACL	"X'04'" .. UP TO 3 EXTRA CLASSES SPECIFIED

OPTION BYTE 2 USAGE INFORMATION:
...GO2OJ: THE SUBMITTED JOB MAY GET ANOTHER JOB NUMBER
WHEN TRANSMITTED TO ANOTHER NODE. NORMALLY THIS JOB
NUMBER IS RETURNED TO THE JOB ORIGINATOR NODE. HOWEVER,
IF THIS BIT IS SET IN THE PUT REQUEST, THE NUMBER AT
THE NODE THE JOB ENTERED THE SYSTEM INITIALLY IS
RETURNED INSTEAD.
THE BIT MUST THEN AGAIN BE SPECIFIED IN THE GCM REQUEST.
...GO2CD: JOB GENERATION MESSAGES CONTAIN THE ID'S, THAT IS,
JOB NAME AND JOB NUMBER OF THE GENERATING AND THE
GENERATED JOB. NORMALLY THE ID OF THE GENERATING JOB IS
USED AS SEARCH CRITERIA FOR THE GCM REQUEST. IF THIS
BIT IS SET, THE ID OF THE GENERATED JOB IS USED.
...GO2WP: IF PEND IS ISSUED WHILE THIS GCM WAIT IS BEING PROCESSED,
ANOTHER GCM-WAIT MAY BE SPECIFIED DURING THE PEND PERIOD,
IF THIS BIT IS SPECIFIED AGAIN.

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(2F)	47	BITSTRING	1	SPLGOPT2	OPTION BYTE 2
		1...		SPLGO2AC	"X'80'" .. CONVERT ASA TO MACHINE CONTROL
		.1..		SPLGO2HU	"X'40'" .. REQ.MATCH TO-UID FOR GEN GET
		..1.		SPLGO2BT	"X'20'" .. IGNORE BLANK TRUNCATION
		...1		SPLGO2QN	"X'10'" .. USE QUEUE RECORD NUMBER
		.... 1...		SPLGO2FE	"X'08'" .. ALLOW TO PUT FE-RECORDS
		.... .1..		SPLGO2OJ	"X'04'" .. PUT: PASS ORG JOB# IN JEM .. GCM: PASS ORG JOB# IN SPL
		.... ..1.		SPLGO2CD	"X'02'" .. GCM: USE GENER'D JOB ID
		1		SPLGO2WP	"X'01'" .. GCM: NEW WAIT AFTER PEND
(30)	48	BITSTRING	1	SPLGEND (0)	SERVES AS ORG POINT FOR OVERLAY
		..11		SPLGSLEN	"*-SPLDS" LENGTH OF GENERAL SECTION

GENERAL SECTION - PART 2

THE FOLLOWING FIELDS CONTAIN DESCRIPTIVE INFORMATION ABOUT
THE QUEUE ENTRY EITHER BUILT OR ACCESSED.

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(30)	48	CHARACTER	1	SPLDDP	DISPOSITION OF QUEUE ENTRY
(31)	49	CHARACTER	1	SPLDPR	PRIORITY OF QUEUE ENTRY
(32)	50	ADDRESS	2	SPLDOJ#	ORIGINAL JOB NUMBER

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(34)	52	CHARACTER	1	SPLDSID	SYSTEM IDENTIFIER
(35)	53	BITSTRING	1	SPLDMOP	GENERAL OPTION BYTE 1 (NOTE - BITS ARE DEFINED IN DMB AND QUEUE RECORD)
		..1.		SPLDMNCS	"X'20'" ..NO COPY SEPARATOR PAGES
		...1		SPLDMOHP	"X'10'" ..HOLD WHEN PRT/PUNCH FAILS
(36)	54	BITSTRING	1	SPLDFLG	GENERAL FLAG BYTE
		1...		SPLDFVSE	"X'80'" ..OUTPUT GENERATED BY VSE SYSTEM
		.1..		SPLDFCKI	"X'40'" ..EXT CKP INFO EXISTS
		..1.		SPLDFCKE	"X'20'" ..EXT CKP INFO NOT AVAILABLE (LOST DUE TO I/O ERROR)
		...1		SPLDSKIP	"X'10'" ..SET SKIP=YES IN AUTOSTART
		.... 1...		SPLDFRUN	"X'08'" ..*$$ JOB NORUN=IGN
(37)	55	ADDRESS	1	SPLDCCPY	CHECKPOINT COPY NUMBER
(38)	56	ADDRESS	4	SPLDRCT	TOTAL RECORD COUNT
(3C)	60	ADDRESS	4	SPLDPCT	TOTAL PAGE COUNT (LST ONLY)
(40)	64	ADDRESS	4	SPLDLCT	CARD/LINE COUNT (LST/PUN ONLY)
(44)	68	ADDRESS	4	SPLDCREC	CHECKPOINT RECORD NUMBER OR PUT-OPEN-RESTART RECORD NUMBER
(48)	72	CHARACTER	16	SPLDUI	USER INFORMATION
(58)	88	CHARACTER	8	SPLDONN	ORIGINATOR NODE NAME
(60)	96	CHARACTER	8	SPLDOUID	ORIGINATOR USER/REMOTE IDENTIFIER
(68)	104	CHARACTER	8	SPLDTNN	TARGET NODE NAME
(70)	112	CHARACTER	8	SPLDTUID	TARGET USER/REMOTE IDENTIFIER
(78)	120	CHARACTER	20	SPLDPRGN	PROGRAMMER NAME
(8C)	140	CHARACTER	8	SPLDROOM	ROOM NUMBER
(94)	148	CHARACTER	8	SPLDDEPT	DEPARTMENT NUMBER
(9C)	156	CHARACTER	8	SPLDBLDG	BUILDING NUMBER
(A4)	164	ADDRESS	2	SPLDLREC	MAXIMUM RECORD LENGTH
		OUTPUT SECTION			
		THE FOLLOWING FIELDS ARE ONLY APPLICABLE FOR EITHER SPOOLING OR RETRIEVING OUTPUT TO/FROM THE LST OR PUN QUEUE.			
(A6)	166	BITSTRING	1	SPLORCFM	RECORD FORMAT
		1...		SPLORSCS	"X'80'" .. SCS PRINT
		.1..		SPLORBMS	"X'40'" .. BMS MAPPING
		..1.		SPLOR327	"X'20'" .. 3270 FORMAT
		...1		SPLORAPA	"X'10'" .. CPDS DATA STREAM (APA DATA)
		.... 1...		SPLORESC	"X'08'" .. ESCAPE MODE
		.... .1..		SPLORASA	"X'04'" .. ASA CONTROL CHARACTER
		.... ..1.		SPLORMCC	"X'02'" .. MACHINE CONTROL CHARACTER
(A7)	167	BITSTRING	1	SPLONCPY	NUMBER OF COPIES = 1
(A8)	168	CHARACTER	4	SPLOCOMP	COMPACTION TABLE NAME (RJE,SNA ONLY)
(AC)	172	CHARACTER	8	SPLOFORM	FORMS IDENTIFIER (FNO)
(B4)	180	CHARACTER	8	SPLOEWTR	SUBSYSTEM NAME
(BC)	188	CHARACTER	8	SPLOFCB	FCB NAME
(C4)	196	CHARACTER	8	SPLOUCB	UCB NAME
(CC)	204	BITSTRING	1	SPLOUCBO	UCB OPTION BYTE
		1...		SPLOUCBD	"X'80'" .. BLOCK DATA CHECK OPTION
		.1..		SPLOUCBF	"X'40'" .. FOLD OPTION

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(CD)	205	BITSTRING	1	SPLONSEP	NUMBER OF SEPARATOR PAGES/CARDS
(CE)	206	BITSTRING	2		UNUSED
		3800 SECTION			
		THE FOLLOWING FIELDS ARE ONLY APPLICABLE FOR 3800 OUTPUT			
(D0)	208	CHARACTER	16	SPL3TAB (0)	CHARACTER ARRANGEMENT TABLES
(D0)	208	CHARACTER	4	SPL3TAB1	CHARACTER ARRANGEMENT TABLE 1
(D4)	212	CHARACTER	4	SPL3TAB2	CHARACTER ARRANGEMENT TABLE 2
(D8)	216	CHARACTER	4	SPL3TAB3	CHARACTER ARRANGEMENT TABLE 3
(DC)	220	CHARACTER	4	SPL3TAB4	CHARACTER ARRANGEMENT TABLE 4
(E0)	224	CHARACTER	4	SPL3MODF	COPY MODIFICATION NAME
(E4)	228	CHARACTER	4	SPL3CCHR	CHAR ARRANGEMENT TABLE FOR COPY MOD
(E8)	232	BITSTRING	8	SPL3CPYG (0)	COPY GROUPINGS
(E8)	232	BITSTRING	1	SPL3CPG1	COPY GROUP 1
(E9)	233	BITSTRING	1	SPL3CPG2	COPY GROUP 2
(EA)	234	BITSTRING	1	SPL3CPG3	COPY GROUP 3
(EB)	235	BITSTRING	1	SPL3CPG4	COPY GROUP 4
(EC)	236	BITSTRING	1	SPL3CPG5	COPY GROUP 5
(ED)	237	BITSTRING	1	SPL3CPG6	COPY GROUP 6
(EE)	238	BITSTRING	1	SPL3CPG7	COPY GROUP 7
(EF)	239	BITSTRING	1	SPL3CPG8	COPY GROUP 8
(F0)	240	CHARACTER	4	SPL3FLSH	FLASH IDENTIFIER
(F4)	244	ADDRESS	1	SPL3FLCT	FLASH COUNT = 255
(F5)	245	BITSTRING	1	SPL3FLG1	FLAG BYTE 1
		1...		SPL3F1BR	"X'80'" .. BURST IS REQUESTED
		.1..		SPL3F1TR	"X'40'" .. 1ST BYTE CONTAINS TRC CHARACTER
		..1.		SPL3F138	"X'20'" .. 3800 SECTION PRESENT
(F6)	246	BITSTRING	2		UNUSED
		EXTENDED SECTION FOR SPL VERSION 2			
(F8)	248	ADDRESS	2	SPLEOPOF	OFFSET TO START OF OPTBS
(FA)	250	ADDRESS	2	SPLEOPLN	LENGTH OF SPECIFIED OPTBS
		EXTENDED SECTION FOR SPL VERSION 3 IR			
(FC)	252	CHARACTER	8	SPLXDIST	DISTRIBUTION CODE
(104)	260	ADDRESS	2	SPLXQRJ#	ORIGINAL RDR JOB NUMBER
(106)	262	ADDRESS	2	SPLXCKIL	EXT CKP INFO LENGTH
(108)	264	ADDRESS	4	SPLXQNUM	QUEUE ENTRY NUMBER
(10C)	268	BITSTRING	1	SPLXFLG1	EXTENDED FLAG BYTE 1
		1...		SPLX1LGN	"X'80'" .. LOG=NO SPECIFIED
		.1..		SPLX1EMG	"X'40'" .. EOJMSG=YES SPECIFIED
		..1.		SPLX1ACE	"X'20'" .. ENTRY CREATED BY J PUN
(10D)	269	BITSTRING	1	SPLXOB1	EXTENDED OPTION BYTE 1
		1		SPLXO1CQ	"X'01'" .. PUT:JEM TO COM.QUEUE
		.... ..1.		SPLXO1DQ	"X'02'" .. PUT:JEM TO COM+USR.QUEUE
(10E)	270	BITSTRING	2	SPLXWAIT	GCM: WAIT TIME (0..27962 S)
		BITSTRING		SPLXWETR	"X'FFFF'" .. GCM: WAIT INDEFINITELY
(110)	272	CHARACTER	8	SPLXSID	SECURITY USERID
(118)	280	CHARACTER	8	SPLXSPW	SECURITY PASSWORD
(120)	288	CHARACTER	8	SPLXPMDE	PROCESSING MODE (PRMODE)
(128)	296	BITSTRING	8	SPLXPRIV	PUT:PRIVATE USER INFO

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(130)	304	CHARACTER	12		FREE FOR FUTURE USE
		EXPRESSION		SPLEOPST	"*-SPLDS" POSSIBLE START OF OPTBS
		EXPRESSION		SPLTLEN	"*-SPLDS" TOTAL LENGTH OF SPL
	OPTB AREA				
(13C)	316	CHARACTER	1	SPLEOPTB (0)	START OF OPTB AREA
	VSE/POWER COMMAND SECTION				
			THE FOLLOWING SECTION IS AN OVERLAY OF THE LAST 3 SECTIONS AND DEFINES THE COMMAND AREA USED WHEN PASSING A VSE/POWER COMMAND.		
(30)	48	CHARACTER	72	SPLCFLD	COMMAND FIELD
		EXPRESSION		SPLGLEN	"*-SPLDS" LENGTH OF CONTROL BLOCK

•User Data in XPCCB Changed by POWER

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(0)	0	BITSTRING	1	PXPBTYP	BUFFER TYPE
		1		PXPBTSPL	"X'01'" .. SPOOL PARAMETER LIST
		.... ..1.		PXPBTNDB	"X'02'" .. NORMAL DATA BUFFER
		11		PXPBTMSG	"X'03'" .. MESSAGE BUFFER
		.... .1..		PXPBTCTL	"X'04'" .. CONTROL RECORD BUFFER
		.... .1.1		PXPBTOPT	"X'05'" .. BUFFER WITH OUTPUT PARAMETER TEXT BLOCK(S)
(1)	1	BITSTRING	1	PXPACT1	ACTION TYPE 1
(2)	2	BITSTRING	1		RESERVED
(3)	3	BITSTRING	1	PXPINFO	USER INFORMATION BYTE
		1...		PXPIMSG	"X'80'" .. MESSAGE(S) QUEUED
		.1..		PXPIORD	"X'40'" .. ORDER PENDING
		..1.		PXPIPSH	"X'20'" .. VSE/POWER IN SHUTDOWN
(4)	4	BITSTRING	1	PXPRETCD	RETURN CODE
		....		PXPRCOK	"X'00'" .. OK, NO ERROR
		.... .1..		PXPRCOKF	"X'04'" .. REQUEST NOT HANDLED
		.... 1...		PXPRCERR	"X'08'" .. REQUEST REJECTED
		.... 11..		PXPRCPVL	"X'0C'" .. PROTOCOL VIOLATED OR SEVERE ERROR
		...1		PXPRCNOC	"X'10'" .. CONNECTION TERMINATED
(5)	5	BITSTRING	1	PXPFBKCD	FEEDBACK CODE
		....		PXP00OK	"X'00'" .. OK, NO ERROR
		1		PXP00EOD	"X'01'" .. END OF DATA
		.... ..1.		PXP00NJB	"X'02'" .. JOB NOT ON JOB BOUNDARY
		11		PXP00NRS	"X'03'" .. NO RECORD SPOOLED
		.... .1..		PXP00RTR	"X'04'" .. RECORD EXCEEDS SPEC. MAX. LENGTH
		.... .1.1		PXP00ZBF	"X'05'" .. ZERO DATA BUFFER
		.... .11.		PXP00CIA	"X'06'" .. CHECKPOINT ID ALTERED
		.... .111		PXP00NCM	"X'07'" .. NO JOB CMPL.MS.RETR.(PUT)
		.... 1...		PXP00LCM	"X'08'" .. 2..5 CMPL.MSG'S (PUT)
		.... 1..1		PXP00OCM	"X'09'" .. 0..1 CMPL.MSG'S (PUT)
		1		PXP04NOF	"X'01'" .. JOB/OUTPUT NOT FOUND, FIND REASONS IN PXPFBKC2
		.... ..1.		PXP04JOP	"X'02'" .. JOB/OUTPUT PROTECTED
		11		PXP04BSY	"X'03'" .. JOB/OUTPUT MARKED ACTIVE (BUSY)
		.... .1..		PXP04NDS	"X'04'" .. JOB/OUTPUT NOT DISPATCHABLE
		.... .1.1		PXP04IDP	"X'05'" .. APPEND ERROR, INVALID DISPOSITION
		.... .11.		PXP04RER	"X'06'" .. RESTART ERROR, OUTSIDE RANGE
		.... .111		PXP04CER	"X'07'" .. CHECKPOINT ERROR, OUTSIDE RANGE
		.... 1...		PXP04SOD	"X'08'" .. SHORT ON SPOOL FILE SPACE (SOD)
		.... 1..1		PXP04SOA	"X'09'" .. SHORT ON ACCOUNT FILE SPACE (SOA)
		.... 1.1.		PXP04BER	"X'0A'" .. REQUEST PROHIBITED IN BROWSE MODE
		.... 1.11		PXP04DNF	"X'0B'" .. NOTHING TO DISPLAY IN QUEUE(S) FIND REASONS IN PXPFBKC2

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
		.... 11..		PXP04TQN	"X'0C'" .. TEMP QUEUE SET NOT FOUND
		.... 11.1		PXP04NMU	"X'0D'" .. NO MATCHING USER ID
		.... 111.		PXP04WDP	"X'0E'" .. RESTART DISP NOT D,H,K,L OR X
		 1111		PXP04JSR	"X'0F'" .. JOB SUFFIX NUMBER MANDATORY
		...1		PXP04NOQ	"X'10'" .. NO ORDER/SIGNAL QUEUED
		...1 ...1		PXP04ONF	"X'11'" .. OPTB(S) NOT FOUND
		...1 ..1.		PXP04NJC	"X'12'" .. NO JOB CMPL.MS.RETR.(GCM)
		...1 ..11		PXP04CKN	"X'13'" .. NO EXTENDED CKP INFO EX.
		...1 .1..		PXP04CKE	"X'14'" .. NO EXTENDED CKP INFO AVAILABLE (LOST DUE TO I/O-ERROR)
		...1 .1.1		PXP04NCK	"X'15'" .. NO CKP INFO EXISTS (NO RECORD/COPY NUMBER/ EXTENDED INFO)
		...1 .11.		PXP04NMF	"X'16'" .. NO JOB CMPL.MS.FOUND(GCM)
		1		PXP08SPL	"X'01'" .. INVALID SPL
		.... ..1.		PXP08REQ	"X'02'" .. UNKOWN REQUEST TYPE
		11		PXP08SRQ	"X'03'" .. UNKOWN SUB-REQUEST TYPE
		.... .1..		PXP08FB2	"X'04'" .. UNKOWN FUNCTION BYTE 2
		.... .1.1		PXP08JNM	"X'05'" .. INVALID JOB NAME
		.... .11.		PXP08QID	"X'06'" .. INVALID QUEUE IDENTIFIER
		.... .111		PXP08CLS	"X'07'" .. INVALID CLASS
		.... 1...		PXP08PWD	"X'08'" .. INVALID PASSWORD
		.... 1..1		PXP08UID	"X'09'" .. INVALID USER/REMOTE-ID
		.... 1.1.		PXP08RFM	"X'0A'" .. INVALID RECORD FORMAT
		.... 1.11		PXP08DSP	"X'0B'" .. INVALID DISPOSITION
		.... 11..		PXP08PRY	"X'0C'" .. INVALID PRIORITY
		.... 11.1		PXP08SID	"X'0D'" .. INVALID SYTEM IDENTIFIER
		.... 111.		PXP08TNN	"X'0E'" .. INVALID DESTINATION NODE
		 1111		PXP08TUN	"X'0F'" .. INVALID DEST. USER/REMOTE
		...1		PXP08FNO	"X'10'" .. INVALID FORMS IDENTIFIER
		...1 ...1		PXP08FCB	"X'11'" .. INVALID FCB NAME
		...1 ..1.		PXP08UCB	"X'12'" .. INVALID UCB NAME
		...1 .1..		PXP08FLH	"X'14'" .. INVALID FLASH IDENTIFIER
		...1 .1.1		PXP08CPT	"X'15'" .. INV. COMPACTION TABLE NAME
		...1 .11.		PXP08CGP	"X'16'" .. INVALID COPY GROUPINGS
		...1 .111		PXP08CHR	"X'17'" .. INVALID CHAR TABLE(S)
		...1 1...		PXP08MOD	"X'18'" .. INV. COPY MODIFICATION NAME
		...1 1..1		PXP08CCR	"X'19'" .. INVALID CHAR FOR COPY MOD
		...1 1.1.		PXP08BTS	"X'1A'" .. BUFFER TOO SMALL
		...1 1.11		PXP08IAO	"X'1B'" .. WRONG SPEC. OF APPEND/RESTART OPT
		...1 11..		PXP08IAB	"X'1C'" .. INVALID ACTION REQUEST
		...1 11.1		PXP08ICR	"X'1D'" .. INVALID CONTROL RECORD
		...1 111.		PXP08PRG	"X'1E'" .. INVALID PROGRAMMER NAME
		...1 1111		PXP08ROO	"X'1F'" .. INVALID ROOM NUMBER
		..1.		PXP08DPT	"X'20'" .. INVALID DEPARTMENT NUMBER
		..1. ...1		PXP08BLD	"X'21'" .. INVALID BUILDING NUMBER
		..1. ..1.		PXP08CON	"X'22'" .. CONFLICTING SPECIFICATIONS
		..1. ..11		PXP08ROL	"X'23'" .. RECEIVED RECORD TOO LARGE
		..1. .1..		PXP08IBT	"X'24'" .. INVALID BUFFER TYPE
		..1. .1.1		PXP08ROS	"X'25'" .. REQUEST OUT OF SEQUENCE
		..1. .11.		PXP08SOS	"X'26'" .. SPL RECEIVED OUT OF SEQUENCE
		..1. .111		PXP08BOS	"X'27'" .. RECEIVED BUFFER OUT OF SEQUENCE
		..1. 1...		PXP08RPH	"X'28'" .. REQUEST PROHIBITED
		..1. 1..1		PXP08ISS	"X'29'" .. INVALID SIGNAL SPECIFICATION OR SIGNAL OUT OF SEQUENCE
		..1. 1.1.		PXP08RPW	"X'2A'" .. RECORD PREFIX WRONG
		..1. 1.11		PXP08FB1	"X'2B'" .. UNKOWN FUNCTION BYTE 1
		..1. 11..		PXP08IML	"X'2C'" .. INVALID MAX. RECORD LENGTH IN SPL
		..1. 11.1		PXP08IEX	"X'2D'" .. INVALID SUBSYSTEM NAME
		..1. 111.		PXP08SPA	"X'2E'" .. COMPLETE RECORD NOT IN BUFFER
		..1. 1111		PXP08ICC	"X'2F'" .. INVALID CARRIAGE CONTROL CHAR
		..11		PXP08IOR	"X'30'" .. INVALID ORDER
		..11 ...1		PXP08JNO	"X'31'" .. INVALID JOB NUMBER(=0)
		..11 ..1.		PXP08JSF	"X'32'" .. INVALID JOB SUFFIX NO (>127)
		..11 ..11		PXP08IUI	"X'33'" .. INVALID USER INFORMATION

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
		..11 .1..		PXP08IPD	"X'34'" .. GET-SPL FROM RDR QUEUE OR PUT-SPL NOT ALLOWED FOR DST
		..11 .1.1		PXP08UXR	"X'35'" .. UNEXPECTED RESPONSE RECEIVED
		..11 .11.		PXP08WOS	"X'36'" .. WAIT FOR ORDER OUT OF SEQUENCE
		..11 .111		PXP08NSP	"X'37'" .. INVALID SEPARATOR PAGES/CARDS
		..11 1...		PXP08IRR	"X'38'" .. INVALID REQUEST FOR RDR
		..11 1..1		PXP08IOP	"X'39'" .. INVALID OPTB SPECIFIED
		..11 1.1.		PXP08OLM	"X'3A'" .. OPTB LENGTH MISMATCH
		..11 1.11		PXP08DOP	"X'3B'" .. DUPLICATE OPTBS SPEC.
		..11 11..		PXP08OTL	"X'3C'" .. SPECIFIED OPTBS TOO LONG
		..11 11.1		PXP08IDH	"X'3D'" .. INVALID DSHR FOUND
		..11 111.		PXP08DIS	"X'3E'" .. INVALID DISTRIBUTION CODE
		..11 1111		PXP08INK	"X'3F'" .. INVALID KEYWORD (SYNTAX)
		.1..		PXP08NDK	"X'40'" .. DEFINE MISS. FOR KEYWORD
		.1.. ...1		PXP08IDV	"X'41'" .. INVALID KEYWORD VALUE
		.1.. ..1.		PXP08CKZ	"X'42'" .. EXT. CKP INFO LENGTH = 0
		.1.. ..11		PXP08CKL	"X'43'" .. EXT. CKP INFO TOO LARGE
		.1.. .1..		PXP08IQN	"X'44'" .. QUEUE RECORD NO INVALID
		.1.. .1.1		PXP08GJN	"X'45'" .. GENERIC JOBNAME
		.1.. .11.		PXP08SEU	"X'46'" .. INVALID SECURITY USERID
		.1.. .111		PXP08SEP	"X'47'" .. INVALID SECURITY PASSWD
		.1.. 1...		PXP08IPM	"X'48'" .. INVALID PROCESSING MODE
		1		PXP0CINS	"X'01'" .. SEND ISSUED, BUT SENDR REQUIRED
		.... ..1.		PXP0CIXF	"X'02'" .. USED XPCC FCT NOT SUPPORTED
		11		PXP0CBTL	"X'03'" .. BUFFER TOO LARGE
		.... .1..		PXP0CPER	"X'04'" .. PROTOCOL ERROR
		.... .1.1		PXP0CPVD	"X'05'" .. PROTOCOL VIOLATION BY DDS (ORDER QUEUED FLAG NOT HONORED)
		.... .111		PXP0CIOE	"X'07'" .. I/O ERROR ON QUEUE/DATA/ACCOUNT FILE
					X'01' .. RESERVED FOR FUTURE USE
		11		PXP10CAA	"X'03'" .. CONNECTION ALREADY ACTIVE
		.... .1.1		PXP10PSP	"X'05'" .. PSTOP GIVEN BY OPERATOR
		.... .11.		PXP10SIE	"X'06'" .. SEVERE INTERNAL ERROR
(6)	6	BITSTRING	2	PXPROFF	OFFSET TO INVALID RECORD
(6)	6	BITSTRING	2	PXPRBLN	REQUIRED BUFFER LENGTH
(6)	6	BITSTRING	2	PXPLEMC	COUNT OF LOST JOB EVNT MSG'S
(6)	6	BITSTRING	1	PXPFBKC2	FEEDBACK CODE 2: ----- part 1 --------- Following valid only with: - PXPRCOKF = PXP04NOF - PXPRCOKF = PXP04DNF
		....		PXPC2OK	"X'00'" .. ALL-CMDS, NO ERROR
		1		PXPC2TEM	"X'01'" .. R|H-CMD NO ACCESS TO DISP=X|A|Y
		.... ..1.		PXPC2NOH	"X'02'" .. H-CMD HOLD ONLY FOR DISP=D|K
		11		PXPC2NOR	"X'03'" .. R-CMD RELEASE ONLY FOR DISP=H|L
		.... .1..		PXPC2NTA	"X'04'" .. A-CMD WARNING NOTHING TO CHANGE
		.... .1.1		PXPC2CPO	"X'05'" .. A-CMD COPY CHANGE FOR ' ' ENTRY BUT ADDITIONAL OPERANDS GIVEN
		.... .11.		PXPC2CDI	"X'06'" .. A-CMD COPY CHANGE FOR ' ' ENTRY BUT 'PDIR' OUTBOUND TASK FOUND
		.... .111		PXPC2CNT	"X'07'" .. A-CMD COPY CHANGE FOR ' ' ENTRY NO SUITABLE ACTIVE TASK FOUND
		.... 1...		PXPC2BAD	"X'08'" .. ALL-CMDS|GET, QUEUE RECORD NOT ACCESSIBLE DUE TO I/O ERROR
		.... 1..1		PXPC2FRE	"X'09'" .. ALL-CMDS|GET, QUEUE REC. EMPTY, ALREADY IN FREE Q-RECORD CHAIN
		.... 1.1.		PXPC2MQU	"X'0A'" .. ALL-CMDS|GET, MISMATCH QUEUE
		.... 1.11		PXPC2MJM	"X'0B'" .. ALL-CMDS|GET, MISMATCH JOB NAME
		.... 11..		PXPC2MJB	"X'0C'" .. ALL-CMDS|GET, MISMATCH JOB NUMB
		.... 11.1		PXPC2IPW	"X'0D'" .. A|H|L|R-CMD, SPL SPECIFIED USER PASSWORD MISMATCHING Q-REC PWD
		.... 111.		PXPC2BPW	"X'0E'" .. A|H|L|R-CMD, DEFAULT SPL PWD NO MATCH TO Q-RECORD PASSWORD
		 1111		PXPC2JFR	"X'0F'" .. A|H|L|R-CMD JOB ONLY, FROM-NODE OR FROM-USER NOT MATCHING OWN

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
		...1		PXPC2OT1	"X'10'" .. A|H|L|R-CMD,OUTPUT ONLY,TO-USER NOT MATCHING TO OWN USER-ID
		...1 ...1		PXPC2OT2	"X'11'" .. A|H|L|R-CMD, SIMILAR PXPC2OT1
		...1 ..1.		PXPC2OT3	"X'12'" .. A|H|L|R-CMD, SIMILAR PXPC2OT1
		...1 ..11		PXPC2OTN	"X'13'" .. A|H|L|R-CMD OUTPUT ONLY,TO-NODE NOT MATCHING TO OWN NODE NAME
		...1 .1..		PXPC2MJS	"X'14'" .. A|H|L|R-CMD|GET, MISMATCHING JOB(OUTPUT) SUFFIX
		...1 .1.1		PXPC2MCL	"X'15'" .. GET-RQ, MISMATCHING JOB CLASS
		...1 .11.		PXPC2MSY	"X'16'" .. GET-RQ, MISMATCH TARGET SYSID
		...1 .111		PXPC2MFU	"X'17'" .. GET-RQ. USERID NOT MATCHING TO 'FROM'-USERID OF JOB ENTRY
		...1 1...		PXPC2MFT	"X'18'" .. GET-RQ. USERID NOT MATCHING TO FROM|TO-USERID OF OUTPUT ENTRY
					----- part 2 ---------
					Following valid only with:
					- PXPRCOKF = PXP08CON
		1		PXPC222A	"X'01'" .. BUFFER LENGTH=0, BUT BUFFER TYPE IS SET
		.... ..1.		PXPC222B	"X'02'" .. BUFFER LENGTH=0, BUT NO ACTION IS SET
		11		PXPC222C	"X'03'" .. BUFFER LENGTH=0, BUT NO ACTION AND NO SIGNAL IS SET, DST TASK
		.... .1..		PXPC222D	"X'04'" .. BUFFER LENGTH=¬0, BUT NO BUFFER TYPE IS SET
		.... .1.1		PXPC222E	"X'05'" .. BUFFER LENGTH=¬0, BUT SIGNAL IS SET, DST TASK
		.... .11.		PXPC222F	"X'06'" .. BUFFER LENGTH=¬0, BUT BUFFER TYPE AND ACTION ARE SET, NO SERVICE IN PROGRESS
		.... .111		PXPC222G	"X'07'" .. BUFFER LENGTH=¬0, BUT BUFFER TYPE AND ACTION ARE SET,GET/CTL/GCM SERVICE IN PROGRESS
		.... 1...		PXPC222H	"X'08'" .. PUT-CLOSE REQUEST, INVALID BUFFER TYPE
		.... 1..1		PXPC222I	"X'09'" .. PUT-SEGMENT REQUEST, INVALID BUFFER TYPE
		.... 1.1.		PXPC222J	"X'0A'" .. PUT-APPEND REQUEST, INVALID BUFFER TYPE
		.... 1.11		PXPC222K	"X'0B'" .. PUT-CHECKPOINT REQUEST, INVALID BUFFER TYPE
		.... 11..		PXPC222L	"X'0C'" .. PUT-QUIT REQUEST, INVALID BUFFER TYPE
					----- part 3 ---------
					Following valid only with:
					- PXPRCOKF = PXP08ROS
		1		PXPC225A	"X'01'" .. BUFFER LENGTH=0, BUT NOT SERVICE AND NO MSG IN PROGRESS, NO VALID REQUEST, NO SIGNAL (DST TASK)
		.... ..1.		PXPC225B	"X'02'" .. GET-SERVICE, SEND DATA REQUEST UBT NO MORE DATA AVAILABLE
		11		PXPC225C	"X'03'" .. MSG-SERVICE, GET MSG REQUEST BUT NO MORE MESSAGES AVAILABLE
		.... .1..		PXPC225D	"X'04'" .. GCM-SERVICE HAS FINISHED< BUT NO NEW SPL RECEIVED
		.... .1.1		PXPC225E	"X'05'" .. GCM-OPEN-KEEP IN PROGRESS, NO GCM-MORE & NO GCM-REMOVE REQUEST
		.... .11.		PXPC225F	"X'06'" .. GCM-OPEN-DELETE IN PROGRESS, NO GCM-MORE REQUEST
		.... .111		PXPC225G	"X'07'" .. GCM-OPEN-REMOVE OR PURGE IN PROGRESS: ANY REQUEST RECEIVED
		.... 1...		PXPUSLN	"*-PXPUSER" LENGTH OF CONTROL BLOCK

•User Data in XPCCB Changed by User

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(0)	0	BITSTRING	1	PXUBTYP	BUFFER TYPE
		1		PXUBTSPL	"X'01'" .. SPOOL PARAMETER LIST
		.... ..1.		PXUBTNDB	"X'02'" .. NORMAL DATA BUFFER

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
		.... .1..		PXUBTCTL	"X'04'" .. CONTROL RECORD BUFFER
(1)	1	BITSTRING	1	PXUACT1	ACTION TYPE 1
		1		PXUATEOD	"X'01'" .. END OF DATA (PUT-FCT)
		.... ..1.		PXUATRQS	"X'02'" .. CLOSE QUEUE ENTRY (GET-FCT)
		11		PXUATABR	"X'03'" .. QUIT-REQUEST
		.... .1..		PXUATSGM	"X'04'" .. SEGMENTATION REQUEST
		.... .1.1		PXUATROE	"X'05'" .. EOD FOR APPENDABLE OUTPUT
		.... .11.		PXUATPRG	"X'06'" .. PURGE QUEUE ENTRY RESQUEST
		.... .111		PXUATCHK	"X'07'" .. CHECKPOINT REQUEST
		.... 1...		PXUATRMR	"X'08'" .. RETURN MESSAGE REQUEST
		.... 1..1		PXUATSDR	"X'09'" .. SEND DATA REQUEST
		.... 1.1.		PXUATFLH	"X'0A'" .. FLUSH HOLD REQUEST
		.... 1.11		PXUATROR	"X'0B'" .. RETURN ORDER/SIGNAL IMMEDIATELY
		.... 11..		PXUATWFR	"X'0C'" .. WAIT TILL ORDER/SIGNAL TO RETURN
		.... 11.1		PXUAT1PF	"X'0D'" .. PRINTING/PUNCHING FAILED
		.... 111.		PXUATCKR	"X'0E'" .. RETRIEVE EXT CKP INFO
		...1		PXUATDEL	"X'10'" .. DELETE RETR. MSG'S (GCM)
		...1 ...1		PXUATGCM	"X'11'" .. RETR. MORE MSG'S (GCM)
(2)	2	BITSTRING	1		RESERVED
(3)	3	BITSTRING	1	PXUINFO	USER INFORMATION BYTE
(4)	4	BITSTRING	1	PXURETCD	RETURN CODE
(5)	5	BITSTRING	1	PXUFBKCD	FEEDBACK CODE
(6)	6	BITSTRING	1	PXUSIGNL	SIGNAL BYTE
		1		PXUSDSTP	"X'01'" .. DEVICE STOPPED
		.... ..1.		PXUSSET	"X'02'" .. SETUP PROCESSED
(7)	7	BITSTRING	1		RESERVED
		.... 1...		PXUUSLN	"*-PXUUSER" LENGTH OF CONTROL BLOCK

•VSE/POWER General Constants

The following statements define some constatns used for the cross partition communication.

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
		..11		SPLMINSZ	"SPLGSLEN" MINIMAL SPL LENGTH
		SIGNED		SPLMAXBS	"64*1024" MAXIMAL BUFFER SIZE FOR POWER

•VSE/POWER Record Prefix Layout

The following statements define the genreal format of each logical data record within a buffer.

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(0)	0	BITSTRING	1	RECCCODE	COMMAND CODE
(1)	1	BITSTRING	1	RECTYPE	RECORD TYPE
		....		RECTNORM	"X'00'" .. NORMAL DATA RECORD
		1		RECTSPL	"X'01'" .. RECORD IS SPOOL PARAMETER LIST
		.... ..1.		RECTFIXM	"X'02'" .. FIXED FORMAT MESSAGE
		11		RECTSEPR	"X'03'" .. SEPARATOR PAGE/CARD RECORD
		.... .1..		RECT3540	"X'04'" .. 3540 DATA RECORD
		.... .1.1		RECTCCR	"X'05'" .. CONTROL COMMAND RECORD
		.... .11.		RECTCPDS	"X'06'" .. CPDS DATA RECORD (APA)
		.... .111		RECTESEP	"X'07'" .. END SEPARATOR PAGE/CARD RECORD
		.... 1...		RECTEOC	"X'08'" .. END OF COPY
		.... 1..1		RECTFJCM	"X'09'" .. FIX. FORM. JOB CMPL MSG
		.... 1.1.		RECTFJGM	"X'0A'" .. FIX. FORM. JOB MESSAGE
(2)	2	SIGNED	2	RECLNGTH	LOGICAL RECORD LENGTH
(4)	4	ADDRESS	4	RECLOGNO	LOGICAL RECORD NUMBER
		.... 1...		RECPRFXL	"*-RECPRFIX" LENGTH OF PREFIX
		.... 1...		RECDATA	"*" DATA RECORD TEXT

•VSE/POWER Fixed Format Queue Display Record

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(0)	0	ADDRESS	2	PXFMRLEN	RECORD LENGTH
(2)	2	BITSTRING	1	PXFMTYPE	RECORD TYPE
		1		PXFMTQDI	"X'01'" .. FIXED FORMAT QUEUE DISPLAY
(3)	3	BITSTRING	1	PXFMVOL	TAPE BAM VOLUME NUMBER
		1...		PXFMVOLA	"X'80'" .. LAST VOLUME FLAG
		.111 1111			"X'7f'" .. (volume number) (127 means 127 or more)
(4)	4	CHARACTER	8	PXFMDATE	DATE
(C)	12	CHARACTER	4	PXFMSTRT	START TIME (0HHMMSSF)
(10)	16	CHARACTER	4	PXFMSTOP	STOP TIME (0HHMMSSF)
(14)	20	CHARACTER	16	PXFMUSER	USER INFORMATION
(24)	36	CHARACTER	8	PXFMNAME	JOB NAME
(2C)	44	ADDRESS	2	PXFMJNUM	JOB NUMBER
(2E)	46	BITSTRING	1	PXFMJSUF	JOB SUFFIX NUMBER
		1...		PXFMJSLA	"X'80'" .. LAST SEGMENT INDICATOR (NOTE: BITS 1 - 7 ARE THE JOB SUFFIX NUMBER(1 - 127) IF ANY)
(2F)	47	CHARACTER	1	PXFMQUID	QUEUE IDENTIFIER (R, L, P)
(30)	48	CHARACTER	1	PXFMCLSS	CLASS
(31)	49	CHARACTER	1	PXFMPRIO	PRIORITY
(32)	50	CHARACTER	1	PXFMDISP	DISPOSITION (' '..IN EXEC.)
(33)	51	ADDRESS	1	PXFMCOPY	NUMBER OF COPIES
(34)	52	BITSTRING	1	PXFMFLG1	CONTROL FLAG 1
		1...		PXFMF1XQ	"X'80'" .. QUEUE SET RESIDES IN XMIT QUEUE
		.1..		PXFMF1AB	"X'40'" .. ABENDED EMTRY, DISP=X
		..1.		PXFMF1AP	"X'20'" .. APPENDABLE ENTRY, DISP=A
		...1		PXFMF1CP	"X'10'" .. CHECKPOINTED CRE-ENTRY
		.... 1...		PXFMF1PF	"X'08'" .. PRT/PUN FAILED ENTRY, D=Y
		.... .111		PXFMF107	"X'04'+X'02'+X'01'" .. RESETS NON-APPLICABLE FLAGS FOR QUEUE RECORD QRS1
		.... .1..		PXFMF1EX	"X'04'" .. DUE DATE EXPIRED
		.... ..1.		PXFMF1SE	"X'02'" .. SECNODE PRESENT
(35)	53	BITSTRING	1	PXFMRCFM	RECORD FORMAT
(36)	54	CHARACTER	1	PXFMSTAT	PAPER STATUS BYTE C'B' .. BURST REQUESTED
(37)	55	CHARACTER	1	PXFMSYID	SYSTEM ID. (TARGET/PROCESS.) OR 'M', IF PARALLEL USERS EXIST ON > 1 SHARING CPU
(38)	56	ADDRESS	4	PXFMREC#	NUMBER OF RECORDS SPOOLED
(3C)	60	ADDRESS	4	PXFMPGE#	NUMBER OF PAGES SPOOLED
(40)	64	ADDRESS	4	PXFMLNE#	NUMBER OF LINES/CARDS SPOOLED
(44)	68	CHARACTER	4	PXFMFLSH	FLASH IDENTIFIER
(48)	72	CHARACTER	8	PXFMFORM	FORMS IDENTIFIER
(50)	80	BITSTRING	8	PXFMCPYG	COPY GROUPINGS
(58)	88	BITSTRING	1	PXFMFLG2	CONTROL FLAG 2 FLAGS REFER TO THE EXECUTION CLASS OF THE SUBJECT JOB FLAGS X'80' - X'04' ARE ONLY SET FOR READER QUEUE ENTRIES, THAT ARE NEITHER IN EXECUTION PREPARATION PHASE NOR IN DISPOSITION = ' ' STATE
		1...		PXFM2SDF	"X'80'" .. CLASS DEFINED AS STATIC
		.1..		PXFM2SRN	"X'40'" .. STATIC CLASS RUNNING

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
		..1.		PXFM2SWW	"X'20'" .. ST. CL. WAITING FOR WORK
		...1		PXFM2DDF	"X'10'" .. CLASS DEFINED AS DYNAMIC
		.... 1...		PXFM2DSP	"X'08'" .. DYNAMIC CLASS SUSPENDED
		.... .1..		PXFM2DEN	"X'04'" .. DYNAMIC CLASS ENABLED
		.... ..1.		PXFM2PRP	"X'02'" .. IN EXECUTION PREP. PHASE
		1		PXFM2RUN	"X'01'" .. *$$ JOB NORUN=IGN
(59)	89	ADDRESS	1	PXFMNSEP	NUMBER OF SEP. PAGES / CARDS
(5A)	90	ADDRESS	2	PXFMJBO#	ORIGINAL JOB NUMBER
(5C)	92	CHARACTER	4	PXFMCMPT	COMPACTION TABLE NAME
(60)	96	CHARACTER	8	PXFMNODE	TARGET DESTINATION NODE NAME
(68)	104	CHARACTER	8	PXFMUSID	TARGET DESTINATION USER/REMOTE ID
(70)	112	CHARACTER	8	PXFMORGN	ORIGINATING NODE NAME
(78)	120	CHARACTER	8	PXFMORGU	ORIGINATING USER/REMOTE ID
(80)	128	CHARACTER	8	PXFMSUBS	SUBSYSTEM NAME (EXTERNAL WRITER ID)
(88)	136	CHARACTER	5	PXFMDDND (0)	NEXT DUE DATE IF NO TES INFO EXISTS OR LST/PUN OUTPUT: ZEROS
(88)	136	CHARACTER	2	PXFMDDN1	DAY OR MONTH
(8A)	138	CHARACTER	1	PXFMDDS1	SEPARATOR /
(8B)	139	CHARACTER	2	PXFMDDN2	DAY OR MONTH

(8D)	141	CHARACTER	5	PXFMDDNT (0)	NEXT DUE TIME
					IF NO TES INFO EXISTS OR LST/PUN OUTPUT: ZEROS
					IF RDR NON-DISPATCHABLE: 2 DASHES (--) AND HEX ZEROS

(8D)	141	CHARACTER	2	PXFMDDNH	HOURS
(8F)	143	CHARACTER	1	PXFMDDS2	SEPARATOR :
(90)	144	CHARACTER	2	PXFMDDNM	MINUTES
(92)	146	CHARACTER	2	PXFMDATC	CENTURY OF CREATION DATE
(94)	148	ADDRESS	4	PXFMQNUM	QUEUE ENTRY NUMBER
(98)	152	CHARACTER	8	PXFMSECN	QUEUE ENTRY SECURITY ZONE (SECNODE)
(A0)	160	CHARACTER	8	PXFMDIST	OUTPUT DISTRIBUTION CODE
(A8)	168	BITSTRING	10	PXFMMACC (0)	MULT. BROWSE ACCESS COUNTS:
(A8)	168	BITSTRING	1	PXFMMACN	.. NON SHARED ACCESS COUNT
(A9)	169	BITSTRING	1	PXFMMAC1	.. SHARED SYSID 1 ACC. CNT.
(AA)	170	BITSTRING	1	PXFMMAC2	.. SHARED SYSID 2 ACC. CNT.
(AB)	171	BITSTRING	1	PXFMMAC3	.. SHARED SYSID 3 ACC. CNT.
(AC)	172	BITSTRING	1	PXFMMAC4	.. SHARED SYSID 4 ACC. CNT.
(AD)	173	BITSTRING	1	PXFMMAC5	.. SHARED SYSID 5 ACC. CNT.
(AE)	174	BITSTRING	1	PXFMMAC6	.. SHARED SYSID 6 ACC. CNT.
(AF)	175	BITSTRING	1	PXFMMAC7	.. SHARED SYSID 7 ACC. CNT.
(B0)	176	BITSTRING	1	PXFMMAC8	.. SHARED SYSID 8 ACC. CNT.
(B1)	177	BITSTRING	1	PXFMMAC9	.. SHARED SYSID 9 ACC. CNT.
		1.11 ..1.		PXFMLENG	"*-PXFMDSCT" LENGTH OF CONTROL RECORD

- VSE/POWER Restart Control Record

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(0)	0	ADDRESS	2	PXRSRLEN	RECORD LENGTH
(2)	2	BITSTRING	1	PXRSTYPE	RECORD TYPE
		.... ..1.		PXRSTRST	"X'02'" .. RESTART CONTROL RECORD
(3)	3	BITSTRING	1		RESERVED FOR FUTURE USE
(4)	4	ADDRESS	4	PXRSRECN	LOG. RECORD NO FROM WHERE TO BEGIN
(8)	8	ADDRESS	1	PXRSRCPY	ASSOCIATED COPY NUMBER
(9)	9	BITSTRING	1	PXRSOPT	OPTION BYTE
		1...		PXRSOPOL	"X'80'" .. POSITIONING ON LINES WANTED
		.1..		PXRSOPAE	"X'40'" .. POSITION AT END OF QUEUE ENTRY, IF REC-NUMB > MAXIMUM
		..1.		PXRSOPOP	"X'20'" .. POSITION ON PAGE WANTED
(A)	10	BITSTRING	2		UNUSED
		.... 11..		PXRSLENG	"*-PXRSDSCT" LENGTH OF RESTART CONTROL RECORD

•VSE/POWER Checkpoint Control Record

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(0)	0	ADDRESS	2	PXCPRLEN	RECORD LENGTH
(2)	2	BITSTRING	1	PXCPTYPE	RECORD TYPE
		11		PXCPTCHK	"X'03'" .. CHECKPOINT CONTROL RECORD
(3)	3	BITSTRING	1	PXCPFLAG	FLAG BYTE
		1...		PXCPFXIE	"X'80'" .. EXTENDED INFO EXISTS
(4)	4	ADDRESS	4	PXCPRECN	LOG. RECORD NUMBER
(8)	8	ADDRESS	1	PXCPRCPY	ASSOCIATED NUMBER OF COPY
(9)	9	BITSTRING	3		UNUSED
		.... 11..		PXCPLENG	"*-PXCPDSCT" LENGTH OF CHECKPOINT CONTROL RECORD
(C)	12	CHARACTER	1	PXCPSTXI (0)	START OF EXTENDED INFO

•VSE/POWER Checkpoint Response Control Record

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(0)	0	ADDRESS	2	PXCRRLEN	RECORD LENGTH
(2)	2	BITSTRING	1	PXCRTYPE	RECORD TYPE
		.... .1..		PXCRTCRS	"X'04'" .. CHECKPOINT RESPONSE CONTROL REC.
(3)	3	BITSTRING	1	PXCRFLAG	FLAG BYTE
		1...		PXCRFXIE	"X'80'" .. EXTENDED INFO EXISTS
		.1..		PXCRFXIS	"X'40'" .. EXTENDED INFO SAVED
(4)	4	CHARACTER	8	PXCRJNAM	JOB NAME
(C)	12	ADDRESS	2	PXCRJNUM	JOB NUMBER
(E)	14	BITSTRING	1	PXCRJSUF	JOB SUFFIX NUMBER
		1...		PXCRJSLA	"X'80'" .. LAST SEGMENT INDICATOR (NOTE: BITS 1 - 7 ARE THE JOB SUFFIX NUMBER(1 - 127) IF ANY)
(F)	15	ADDRESS	1	PXCRRCPY	ASSOCIATED COPY NUMBER
(10)	16	ADDRESS	4	PXCRRECN	LOG. RECORD NUMBER ASSOCIATED WITH CHECKPOINT
(14)	20	ADDRESS	4	PXCRQNUM	QUEUE RECORD NUMBER
(18)	24	ADDRESS	4		RES. FOR FUTURE @D52BDHS
		...1 11..		PXCRLENG	"*-PXCRDSCT" LENGTH OF CHECKPOINT RESPONSE REC.
(1C)	28	CHARACTER	1	PXCRSTXI (0)	START OF EXTENDED INFO

•VSE/POWER GET_OPTB Control Record

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(0)	0	ADDRESS	2	PXGORLEN	RECORD LENGTH
(2)	2	BITSTRING	1	PXGOTYPE	RECORD TYPE
		.... 1...		PXGOTGOP	"X'08'" .. GET_OPTB CONTROL RECORD
(3)	3	BITSTRING	1		RESERVED FOR FUTURE USE
(4)	4	ADDRESS	2	PXGOID	OPTB IDENTIFIER (0 FOR ALL)
		.... .11.		PXGOLENG	"*-PXGODSCT" LENGTH OF GET_OPTB CTL REC.

•VSE/POWER MODIFY_OPTB Control Record

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(0)	0	ADDRESS	2	PXMORLEN	RECORD LENGTH
(2)	2	BITSTRING	1	PXMOTYPE	RECORD TYPE
		.... 1..1		PXMOTMOP	"X'09'" .. MODIFY_OPTB CONTROL REC.
(3)	3	BITSTRING	1		RESERVED FOR FUTURE USE
(4)	4	BITSTRING	4		RESERVED FOR PIPELINING
		.... 1...		PXMOHDRL	"*-PXMODSCT" LENGTH OF FIXED HEADER
(8)	8	CHARACTER	1	PXMOOPTB (0)	OUTPUT PARAMETER TEXT BLOCK

•DSECT for Old Version SPL's

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(0)	0	CHARACTER	48	XTSGSECT	GENERAL SECTION PART 1
(30)	48	CHARACTER	118	XTSDSECT	GENERAL SECTION PART 2
(A6)	166	CHARACTER	42	XTSOSECT	OUTPUT SECTION
(D0)	208	CHARACTER	40	XTS3SECT	3800 SECTION
(F8)	248	CHARACTER	4	XTSESECT (0)	OPTB EXTENSION @D52QDHS
(F8)	248	ADDRESS	2	XTSOPOF	OFFSET OF OPTBS
(FA)	250	ADDRESS	2	XTSOPLN	LENGTH OF OPTBS
		1111 11..		XTSEOPST	"*" POSSIBLE START OF OPTBS
(FC)	252	CHARACTER	1	XTSEOPTB (0)	START OF OPTBS

•VSE/POWER Order Control Records

The Order Control Record consists of two sections:
- Header Section
- Variable Order Data Section

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(0)	0	ADDRESS	2	PORDRLEN	RECORD LENGTH
(2)	2	BITSTRING	1	PORDTYPE	RECORD TYPE
		.... .1.1		PORDREC	"X'05'" .. ORDER CONTROL RECORD
(3)	3	BITSTRING	1	PORDMOD	ORDER REQUEST TYPE
		1		PORDMSTR	"X'01'" .. START DEVICE ORDER
		.... ..1.		PORDMSTP	"X'02'" .. STOP DEVICE ORDER
		11		PORDMRST	"X'03'" .. RESTART DEVICE ORDER
		.... .1..		PORDMPGO	"X'04'" .. REACTIVATE DEVICE ORDER
		.... .1.1		PORDMSET	"X'05'" .. SETUP DEVICE ORDER
		.... .11.		PORDMFLH	"X'06'" .. FLUSH DEVICE ORDER
		.... .111		PORDMXMT	"X'07'" .. USER DEFINED ORDER
		...1		PORDMSND	"X'10'" .. SEND MESSAGE ORDER

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
		...1 ...1		PORDMSLD	"X'11'" .. SET LOGICAL DESTINATION ORDER
		...1 ..1.		PORDMPAO	"X'12'" .. ACCOUNT RECORD ORDER
(4)	4	BITSTRING	1	PORDFLAG	FLAG BYTE
		1...		PORDFQFD	"X'80'" .. QUEUE FOR DISPLAY
(5)	5	ADDRESS	1	PORDMSGL	LENGTH OF MESSAGE
(6)	6	BITSTRING	2	PORDAFPL	LENGTH OF ADVANCED FUNCTION PRINTING ACCOUNT RECORD
(8)	8	CHAR-ACTER	24	PORDDEST (0)	DESTINATION
(8)	8	CHAR-ACTER	8	PORDSUBS	REQUESTING SUBSYSTEM ID
(10)	16	CHAR-ACTER	8	PORDNODE	REQUESTING NODE NAME
(18)	24	CHAR-ACTER	8	PORDUSER	REQUESTING USER IDENTIFIER
		..1.		PORDHLEN	"*-PORDER" LENGTH OF HEADER SECTION
(20)	32	BITSTRING	1	PORDSTRT (0)	SERVES AS ORG POINT FOR OVERLAY
	START DEVICE ORDER SECTION				
(20)	32	CHAR-ACTER	8	PORDSDEV	DEVICE NAME
(28)	40	CHAR-ACTER	4	PORDSCLS	CLASS(ES)
(2C)	44	BITSTRING	2		RESERVED FOR FUTURE USE
(2E)	46	BITSTRING	2	PORDSFLG	FLAG BYTE
		1...		PORDSSKP	"X'80'" .. PSTART WITH SKIP=YES
(2F)	47	ADDRESS	1	PORDSPSL	LENGTH OF PARAMETER STRING
(30)	48	CHAR-ACTER	60	PORDSPRM	PARAMETER STRING
		.11. 11..		PORDSLEN	"*-PORDER" LENGTH OF START DEVICE ORDER
	STOP DEVICE ORDER SECTION				
(20)	32	BITSTRING	1	PORDPTRB	TERMINATION REQUEST BYTE
		1...		PORDPEOJ	"X'80'" .. TERMINATE AT END-OF-JOB
		.1..		PORDPIMM	"X'40'" .. TERMINATE IMMEDIATELY
		..1.		PORDPRST	"X'20'" .. TERMINATE WITH RESTART
(21)	33	BITSTRING	2		RESERVED FOR FUTURE USE
(23)	35	ADDRESS	1	PORDPPSL	LENGTH OF PARAMETER STRING
(24)	36	CHAR-ACTER	60	PORDPPRM	PARAMETER STRING
		.11.		PORDPLEN	"*-PORDER" LENGTH OF STOP DEVICE ORDER
	SETUP DEVICE ORDER SECTION				
(20)	32	ADDRESS	4	PORDUPGE	NUMBER OF PAGES
(24)	36	BITSTRING	11		RESERVED FOR FUTURE USE
(2F)	47	ADDRESS	1	PORDUPSL	LENGTH OF PARAMETER STRING
(30)	48	CHAR-ACTER	60	PORDUPRM	PARAMETER STRING
		.11. 11..		PORDULEN	"*-PORDER" LENGTH OF SETUP DEVICE ORDER
	REACTIVATE DEVICE ORDER SECTION				
(20)	32	BITSTRING	3		RESERVED FOR FUTURE USE
(23)	35	ADDRESS	1	PORDGPSL	LENGTH OF PARAMETER STRING
(24)	36	CHAR-ACTER	60	PORDGPRM	PARAMETER STRING
		.11.		PORDGLEN	"*-PORDER" LENGTH OF REACTIVATE DEVICE SECTION
	RESTART DEVICE ORDER SECTION				
(20)	32	BITSTRING	1	PORDTFLG	FLAG BYTE
		1...		PORDTPOS	"X'80'" .. POSITIVE DISPLACEMENT
		.1..		PORDTMIN	"X'40'" .. NEGATIVE DISPLACEMENT
		..1.		PORDTABS	"X'20'" .. DISPLACEMENT FROM BEGIN
(21)	33	BITSTRING	3		RESERVED FOR FUTURE USE
(24)	36	ADDRESS	4	PORDTPGE	NUMBER OF PAGES / LINES

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(28)	40	BITSTRING	7		RESERVED FOR FUTURE USE
(2F)	47	ADDRESS	1	PORDTPSL	LENGTH OF PARAMETER STRING
(30)	48	CHARACTER	60	PORDTPRM	PARAMETER STRING
		.11. 11..		PORDTLEN	"*-PORDER" LENGTH OF RESTART DEVICE SECTION
FLUSH DEVICE ORDER SECTION					
(20)	32	BITSTRING	1	PORDFFLG	FLAG BYTE
		1...		PORDFHLD	"X'80'" .. FLUSH HOLD REQUESTED
(21)	33	BITSTRING	2		RESERVED FOR FUTURE USE
(23)	35	ADDRESS	1	PORDFPSL	LENGTH OF PARAMETER STRING
(24)	36	CHARACTER	60	PORDFPRM	PARAMETER STRING
		.11.		PORDFLEN	"*-PORDER" LENGTH OF FLUSH DEVICE SECTION
XMIT DEVICE ORDER SECTION					
(20)	32	ADDRESS	1	PORDXPSL	LENGTH OF COMMAND
(21)	33	CHARACTER	132	PORDXPRM	USER DEFINED COMMAND
		1.1. .1.1		PORDXLEN	"*-PORDER" LENGTH OF XMIT DEVICE SECTION
		1.1. .1.1		PORDERMX	"PORDXLEN" MAXIMAL ORDER LENGTH
SEND MESSAGE ORDER SECTION (INBOUND)					
(20)	32	CHARACTER	120	PORDMMSG	MESSAGE TEXT
		1..1 1...		PORDMLEN	"*-PORDER" LENGTH OF SEND MESSAGE ORDER SECTION
SET LOGICAL DESTINATION ORDER SECTION (INBOUND)					
(20)	32	CHARACTER	64	PORDLOG8 (0)	8 LOGICAL DESTINATION NAMES
(20)	32	CHARACTER	8	PORDDLOG (8)	LOGICAL DESTINATION NAME
		.11.		PORDDLEN	"*-PORDER" LENGTH OF SET LOG. DEST. SECTION
PUT ACCOUNT RECORD ORDER (INBOUND)					
(20)	32	BITSTRING	1	PORDAFPA (0)	ADVANCED FUNCTION PRINTING ACCOUNT RECORD

- VSE/POWER Order Response Control Record

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(0)	0	ADDRESS	2	PORSRLEN	RECORD LENGTH
(2)	2	BITSTRING	1	PORSTYPE	RECORD TYPE
		.... .11.		PORSREC	"X'06'" .. ORDER RESPONSE CONTROL RECORD
(3)	3	BITSTRING	1	PORSMOD	ORDER REQUEST TYPE SEE ORDER RECORD FOR DEFINITIONS
(4)	4	BITSTRING	1	PORSFLAG	FLAG BYTE
		1...		PORSFMID	"X'80'" .. TEXT CONTAINS MSG ID(DOM)
(5)	5	ADDRESS	1	PORSMSGL	LENGTH OF MESSAGE
(6)	6	BITSTRING	2	PORSRCFC (0)	RETURN- AND FEEDBACK CODE
(6)	6	BITSTRING	1	PORSRETC	ORDER RESPONSE RETURN CODE
		....		PORSROK	"X'00'" .. ORDER ACCEPTED
		.... .1..		PORSROKF	"X'04'" .. ORDER ACCEPTED BUT REQ. CAN NOT BE HANDLED
		.... 1...		PORSRINV	"X'08'" .. ORDER INVALID OR NOT ACCEPTED
(7)	7	BITSTRING	1	PORSFDBK	ORDER RESPONSE FEEDBACK CODE
		....		PORSFOK	"X'00'" .. OK FEEDBACKCODE FROM USER TO POWER
		1		PORSFPAR	"X'01'" .. PARM STRING MISSING OR INVALID
		.... ..1.		PORSFONA	"X'02'" .. ORDER NOT ACCEPTED
		11		PORSFDUN	"X'03'" .. PSTART - DEVICE UNKNOWN
		.... .1..		PORSFDBS	"X'04'" .. PSTART - DEVICE BUSY

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
		.... .1.1		PORSFDOS	"X'05'" .. PSTART - DEVICE OUT OF SERVICE
		.... .11.		PORSFDRJ	"X'06'" .. PSTART - REJECTED FEEDBACKCODE FROM POWER TO USER
		1		PORSFNAC	"X'01'" .. ACCNTING NOT INITIALIZED
		1		PORSFINV	"X'01'" .. INVALID OR UNKNOWN ORDER
		.... ..1.		PORSFOTS	"X'02'" .. ORDER TOO SHORT
		11		PORSFMSG	"X'03'" .. MSG LENGTH TOO LARGE
		.... .1..		PORSFSLD	"X'04'" .. SLD WITH INVALID DESTINATIONS
		.... .1.1		PORSFPAC	"X'05'" .. LENGTH FIELDS MISMATCH
		.... .11.		PORSFRTL	"X'06'" .. ACCNT REC TOO SMALL/LG
(8)	8	CHARACTER	24	PORSDEST (0)	DESTINATION
(8)	8	CHARACTER	8	PORSSUBS	DESTINATION SUBSYSTEM ID
(10)	16	CHARACTER	8	PORSNODE	DESTINATION NODE NAME
(18)	24	CHARACTER	8	PORSUSER	DESTINATION USER IDENTIFIER
		..1.		PORSHLEN	"*-PORDRESP" LENGTH OF HEADER SECTION
(20)	32	CHARACTER	120	PORSMSG	MESSAGE TEXT
(20)	32	BITSTRING	4	PORSMID	MESSAGE ID FOR DOM
		..1. .1..		PORSMLEN	"*-PORDRESP" LENGTH OF SHORT RECORD
		1..1 1...		PORSTLEN	"*-PORDRESP" LENGTH OF TOTAL RECORD

•VSE/POWER Signal Control Record

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(0)	0	ADDRESS	2	PSGNRLEN	RECORD LENGTH
(2)	2	BITSTRING	1	PSGNLTYP	RECORD TYPE
		.... .111		PSGNLREC	"X'07'" .. SIGNAL CONTROL RECORD
(3)	3	BITSTRING	1	PSGNLMOD	SIGNAL TYPE
		1		PSGNLTOA	"X'01'" .. OUTPUT ARRIVED SIGNAL
(4)	4	BITSTRING	4		RESERVED
		.... 1...		PSGNLLEN	"*-PSIGNAL" LENGTH OF TOTAL RECORD

•VSE/POWER Message Control Record

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(0)	0	ADDRESS	2	PMSGRLEN	RECORD LENGTH
(2)	2	BITSTRING	1	PMSGTYPE	RECORD TYPE
		1...		PMSGTREC	"X'80'" .. MESSAGE CONTROL RECORD
(3)	3	BITSTRING	1		RESERVED
(4)	4	BITSTRING	1	PMSGFLAG	FLAG BYTE
(5)	5	ADDRESS	1	PMSGTXTL	MESSAGE LENGTH
(6)	6	BITSTRING	2		RESERVED FOR FUTURE USE
(8)	8	CHARACTER	8	PMSGSUBS	DESTINATION SUBSYSTEM ID
(10)	16	CHARACTER	8	PMSGNODE	DESTINATION NODE NAME
(18)	24	CHARACTER	8	PMSGUSER	DESTINATION USER IDENTIFIER
		..1.		PMSGHLEN	"*-PMSGREC" LENGTH OF HEADER SECTION
(20)	32	CHARACTER	120	PMSGTEXT	MESSAGE TEXT
		1..1 1...		PMSGTLEN	"*-PMSGREC" LENGTH OF TOTAL RECORD

•VSE/POWER Notify Control Record

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(0)	0	ADDRESS	2	PNTYRLEN	RECORD LENGTH
(2)	2	BITSTRING	1	PNTYTYPE	RECORD TYPE
		1... ...1		PNTYTREC	"X'81'" .. NOTIFY CONTROL RECORD
(3)	3	BITSTRING	1		RESERVED
(4)	4	BITSTRING	1	PNTYFLAG	FLAG BYTE
(5)	5	BITSTRING	3		RESERVED FOR FUTURE USE
(8)	8	CHARACTER	8	PNTYJNAM	JOB NAME
(10)	16	ADDRESS	2	PNTYJNUM	JOB NUMBER
(12)	18	BITSTRING	1	PNTYJSUF	JOB SUFFIX
		1...		PNTYJSLA	"X'80'" .. LAST SEGMENT INDICATOR (NOTE: BITS 1 - 7 ARE THE JOB SUFFIX NUMBER(1 - 127) IF ANY)
(13)	19	CHARACTER	1	PNTYJCLA	CLASS
(14)	20	CHARACTER	8	PNTYDEST	TARGET USER IDENTIFIER
		...1 11..		PNTYLEN	"*-PNTYREC" LENGTH OF CONTROL RECORD

•Class of Job Event Messages: Fixed Format Job Completion Message Record (JCM Record)

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(0)	0	CHARACTER	5	JCMID	MESSAGE IDENTIFIER (1Q5DI)
(5)	5	CHARACTER	2		RESERVED FOR FUTURE USE
(7)	7	BITSTRING	1	JCMFLT	SYSTEM CONFIG INFO (COMREG)
		1...		JCMFDD	"X'80'" ..DATE FORMAT IS DDMMYY
(8)	8	CHARACTER	8	JCMFNAM	JOB NAME
(10)	16	BITSTRING	4	JCMFNUM	JOB NUMBER (JOB NUM)
(14)	20	BITSTRING	4	JCMFONUM	GENERAT'G JOB NUM (ORG.JOB)
(18)	24	CHARACTER	8	JCMFNOD	PROCESSING NODE NAME
(20)	32	CHARACTER	8	JCMFECT	PROCESSING COMPLETION TIME
(28)	40	CHARACTER	4	JCMFLRC	LAST RETURN CODE
(2C)	44	CHARACTER	4	JCMFMRC	HIGHEST RETURN CODE
(30)	48	CHARACTER	8	JCMFECD	PROCESSING COMPLETION DATE
(38)	56	BITSTRING	1	JCMFJC7	JOB CNTROL FLAG 7 (JCSW7)
		1...		JCMFJ7CA	"X'80'" ..OPERATOR CANCEL PENDING
		.... ..1.		JCMFJ7JC	"X'02'" ..JOB CONTROL CANCEL
(39)	57	BITSTRING	1	JCMFJC8	JOB CNTROL FLAG 8 (JCSW8)
		.... 1...		JCMFJ8AB	"X'08'" ..ABNORMAL TERMINATION
(3A)	58	CHARACTER	10	JCMFDUR	JOB DURATION HHHH/MM/SS
(44)	68	CHARACTER	2	JCMFECDC	CENTURY OF PROC. COMPLETION
(46)	70	CHARACTER	10		RESERVED FOR FUTURE USE
(50)	80	BITSTRING	8	JCMFPRIV	DATA FROM SPLXPRIV
(58)	88	CHARACTER	8	JCMFUSID	USERID FROM SPLGUS
		.11.		JCMFLEN	"*-JCMDS" LENGTH OF JCM RECORD

•Class of Job Event Messages: Fixed Format Job Generation Message Record (JGM Record)

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(0)	0	CHARACTER	5	JGMID	MESSAGE IDENTIFIER (1Q5HI)
(5)	5	CHARACTER	3		RESERVED FOR FUTURE USE
(8)	8	CHARACTER	8	JGMFNAM	GENERATING JOB NAME
(10)	16	BITSTRING	4	JGMFNUM	GENERATING JOB NUMBER
(14)	20	CHARACTER	8	JGMFNNAM	GENERATED JOB NAME
(1C)	28	BITSTRING	4	JGMFNNUM	GENERATED JOB NUMBER
(20)	32	CHARACTER	44		RESERVED FOR FUTURE USE
(4C)	76	BITSTRING	4	JGMF1NUM	ORIGINAL GEN'TING JOB NUMB
(50)	80	BITSTRING	8	JGMFPRIV	DATA FROM SPLXPRIV
(58)	88	CHARACTER	8	JGMFUSID	USERID FROM SPLGUS
		.11.		JGMFLEN	"*-JGMDS" LENGTH OF JGM RECORD ASM H V 02 11.32

SPL Checking Parameter List

Definition Macro: IPW$SSJ DSECT

The parameter list is used by the parameter checking routine (IPW$$PC).

Bytes Hex.	Label of Field	Description/Function
00	PCPLDS	Start of DSECT
00-03	PCPLISPL	Address of SPL to be checked
04-05		Reserved for future use
06	PCPLRETC	Return code
07	PCPLFBKC	Feedback code
08-0B	PCPLWSPL	Address of work SPL to be updated
0C-0F	PCPLQREC	Address of queue record to be updated
10-13	PCPLJHR	Address of JHR to be updated
14-17	PCPLDHR	Address of DSHR to be updated
18-19		Unused
1A-1B	PCPLSSJC	Save area for carrier subtype
• Work Area		
1C-1D	PCPLSTYP	Work Area to check carrier type
1E	PCPLCTX	Context constellation flags
	PCPLCTCG	X'80' - 3800 copy groupings present
	PCPLCTMC	X'40' - 3800 copy modification CAT present
	PCPLCTMC	X'20' - Security Userid present
	PCPLCTMC	X'10' - Security Password present
1F	PCPLWKF	Work area flags
	PCPLWKF3	X'80' - 3800 section present in SPL
	PCPLWKFH	X'40' - Disposition equal 'H' or 'L'
20	PCPLCTYP	Carrier type
21	PCPLWCGN	Number of 3800 copy groups
22-23		Reserved for future use
24-27	PCPLFFWA	Address of first fixed format work area (FFWA)
28-2B	PCPLWK1	Work area 1
2C-33	PCPLWK2	Work area 2
34-37	PCPLWKJP	Address of VSE/POWER section in JHR
38-3B	PCPLWKDP	Address of VSE/POWER section in DSHR
3C-3F	PCPLWKD3	Address of 3800 section in DSHR
40-43	PCPLWKS1	Link register save area 1
44-47	PCPLWKS2	Link register save area 2
48-4B	PCPLWKS3	Link register save area 3

Spool Access Support Task Work Area

Definition Macro: IPW$DXW

This macro is used to produce a DSECT for the Spool Access Support (SAS) task work area. The work area is retrieved by the SAS master task and released by its user, the SAS task. The work area is anchored into the TCB at label TCBXWRKA. The format is as follows:

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
				Spool Access Support Task Workarea	
(0)	0	DBL WORD	8	XTWAREA (0)	WORK AREA
(0)	0	CHARACTER	48	XTWDD	DYNAMIC DATA DUE TO REENTRANCY FOR MAINLINE MODULE IPW$$XT
(30)	48	CHARACTER	16	XTWSD	SECTION DESCRIPTOR
(40)	64	ADDRESS	4	XTWTCB	ADDRESS OF OWNING TASK
(44)	68	SIGNED	2	XTWTOKEN	TOKEN OF TASK
(46)	70	BITSTRING	1	XTWSECFG	VSE SECURITY FLAG
(47)	71	BITSTRING	1	XTWSECF2	VSE SECURITY TOKEN FLAG
		.... 1...		XTWSECFT	"X'08'" .. SECURITY TOKEN FG TOKTRST
(48)	72	ADDRESS	4	XTWSPARD	SAVED SPOOL ACCOUNT RECORD
(4C)	76	SIGNED	2	XTWSPARL	LENGTH OF SAVED SPOOL ACCOUNT RECORD
(4E)	78	SIGNED	2		NOT USED
(50)	80	BITSTRING	8	XTWUUSER	SAVED RECEIVED USER DATA FROM XPCCB
(58)	88	CHARACTER	8	XTWSECUR	VSE SECURITY USERID
(60)	96	ADDRESS	4		NOT USED
(64)	100	ADDRESS	4		NOT USED
(68)	104	ADDRESS	4		NOT USED
(6C)	108	ADDRESS	4		NOT USED
	WORKAREA FOR A FUNCTION MODULE (IPW$$XTC, IPW$$XTG, IPW$$XTP, IPW$$XTM)				
(70)	112	CHARACTER	380	XTWFAUTD	DYNAMIC DATA DUE TO REENTRANCY FOR IPW$$XTC/G/P/M @D61LDTR
(1EC)	492	CHARACTER	16	XTWFSD	SECTION DESCRIPTOR
(1FC)	508	BITSTRING	1	XTWSTAF1	STATUS BYTE 1
		1...		XTWSF1LO	"X'80'" .. LOGICAL INTERFACE OPENED
		.1..		XTWSF1QP	"X'40'" .. QUEUE ENTRY PROCESSING
		..1.		XTWSF1ED	"X'20'" .. END OF DATA ENCOUNTERED
		...1		XTWSF1LR	"X'10'" .. EOD, BUT STILL 1 REC TO PROCESS
		.... 1...		XTWSF1ER	"X'08'" .. ERROR FOUND
		.... .1..		XTWSF1NR	"X'04'" .. BUFFER NOR RE-USABLE FOR RECEIVE
		.... ..1.		XTWSF1IE	"X'02'" .. INITIAL END OF DATA
(1FD)	509	BITSTRING	1	XTWSTAF2	STATUS BYTE 2
		1...		XTWSF2PC	"X'80'" .. PUT-REQUEST CHECKPOINTED OUTPUT
		.1..		XTWSF2PL	"X'40'" .. PUT-REQUEST LOCATED QUEUE SET
(1FE)	510	BITSTRING	1	XTWFUNRC	RETURN CODE OF FUNCTION ROUTINE
		....		XTWFROK	"X'00'" .. NO ERROR OCCURRED
		1...		XTWFRST	"X'80'" .. STOP TASK
		.1..		XTWFRER	"X'40'" .. END OF REQUEST
		..1.		XTWFRSOA	"X'20'" .. SHORT ON ACCOUNT SITUATION
		.... 1...		XTWFROPN	"X'08'" .. REQUEST OPEN ERROR
		.... .1..		XTWFRMER	"X'04'" .. REQUEST SOD ERROR DURING PUT
		.... ..1.		XTWFRSUB	"X'02'" .. SUBREQUEST ERROR
(1FF)	511	BITSTRING	1	XTWACCSP	CANCEL CODE FOR GET
(200)	512	ADDRESS	4	XTWPLIR0	LOGICAL INTERFACE REG 0
(204)	516	ADDRESS	4	XTWPLIR1	LOGICAL INTERFACE REG 1
(208)	520	SIGNED	4	XTWPUTCK	PUT CHECKPOINT ID
(20C)	524	ADDRESS	4	XTWOPIAD	ADDRESS OF OPI PARALIST
(210)	528	ADDRESS	4		NOT USED

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(214)	532	ADDRESS	4	XTWFFSV (14)	SAVE AREA FOR TCB, REGD, REGE - 9
OVERLAY STRUCTURE OF STATUS BYTE 1 FOR MODULE IPW$$XTM					
(1FC)	508	BITSTRING	1	XTWMSTAT	DEFINE XTM STATUS BYTE
		1		XTWMSKF2	"X'01'" .. SKIP FLAG2 MESSAGES
		.... ..1.		XTWMRTF1	"X'02'" .. RETURN FLAG1 MESSAGE
		11		XTWMRTF2	"X'03'" .. RETURN FLAG2 MESSAGE
		.... .1..		XTWMRTAL	"X'04'" .. RETURN MESSAGE
		.... .1.1		XTWMDLF2	"X'05'" .. SWITCH OFF FLAG2 MSGS
OVERLAY STRUCTURE OF STATUS BYTE 2 FOR MODULE IPW$$XTM					
(1FD)	509	BITSTRING	1	XTWMTOK	DEFINE XTM TOKEN
		1		XTWMTGM	"X'01'" .. GCM-MORE TOKEN
		.... ..1.		XTWMTGR	"X'02'" .. GCM-REMOVE TOKEN
		11		XTWMTGK	"X'03'" .. GCM-OPEN-KEEP TOKEN
OVERLAY STRUCTURE OF CANCEL CODE BYTE XTWACCSP FOR IPW$$XTM					
(1FF)	511	BITSTRING	1	XTWMFL	DEFINE FLAG BYTE
		1...		XTWMMSR	"X'80'" .. ONE MSG RETRIEVED SO FAR
		.1..		XTWMACI	"X'40'" .. ACIE FOUND
		..1.		XTWMFOU	"X'20'" .. JC MESSAGE FOUND
		...1		XTWMRAF	"X'10'" .. REMOVE ACIE FLAG
		.... 1...		XTWMUTR	"X'08'" .. SPLGUS TRANLATED IN XTM
		.... .1..		XTWMDNF	"X'04'" .. DO NOT FLAG ACIE (PURGE)
		.... ..1.		XTWDUM2	"X'02'" .. RESERVED FOR FUTURE
		1		XTWDUM1	"X'01'" .. RESERVED FOR FUTURE
WORKAREA FOR SUBROUTINE MODULE IPW$$XTS					
(24C)	588	CHARACTER	96	XTWSAUTD	DYNAMIC DATA DUE TO REENTRANCY FOR SUBROUTINE MODULE IPW$$XTS
(2AC)	684	CHARACTER	16	XTWSSD	SECTION DESCRIPTOR
(2BC)	700	BITSTRING	1	XTWSUB	REQUEST FOR SUBROUTINE MODULE
		1		XTWSUBWE	"X'01'" .. WAIT FOR NEXT EVENT
		.... ..1.		XTWSUBRV	"X'02'" .. PROCESS POSTED RECEIVE ECB
		11		XTWSUBRP	"X'03'" .. SEND REPLY
		.... .1..		XTWSUBXE	"X'04'" .. TEST XPCC ERROR
		.... .1.1		XTWSUBSM	"X'05'" .. SEND MESSAGE FOR DST TASK
		.... .11.		XTWSUBJH	"X'06'" .. PROCESS JOB HEADER RECORD
		.... .111		XTWSUBDH	"X'07'" .. PROCESS DATASET HEADER RECORD
(2BD)	701	BITSTRING	1	XTWSUBRC	RETURN CODE OF SUBROUTINE
		....		XTWSROK	"X'00'" .. NO ERROR OCCURRED
		1...		XTWSRST	"X'80'" .. STOP TASK
		.1..		XTWSRPE	"X'40'" .. PROTOCOL ERROR
		..1.		XTWSRRE	"X'20'" .. REQUEST ERROR
		...1		XTWSRBY	"X'10'" .. LINE BUSY
(2BE)	702	BITSTRING	1	XTWSTAS1	STATUS BYTE 1 FOR SUBROUTINE
		1...		XTWSS1ST	"X'80'" .. STOP DUE TO DST TASK
		.1..		XTWSS1OR	"X'40'" .. WAITING FOR ORDER RESPONSE
		..1.		XTWSS1WR	"X'20'" .. WAITING FOR REPLY/REACTIVATION
		...1		XTWSS1OQ	"X'10'" .. RECEIVED ORDER QUEUED
(2BF)	703	BITSTRING	1	XTWSTAS2	STATUS BYTE 2 FOR SUBROUTINE
		1...		XTWSS2SF	"X'80'" .. SECTION FOUND IN CONTROL RECORD
(2C0)	704	BITSTRING	1	XTWSOVER	VERSION OF OLD SPL
(2C1)	705	BITSTRING	1		NOT USED
(2C2)	706	BITSTRING	1		NOT USED
(2C3)	707	BITSTRING	1		NOT USED
(2C4)	708	ADDRESS	4	XTWSCTP	POINTER TO FOUND SECTION
(2C8)	712	ADDRESS	4	XTWRIORD	REPLY AREA FOR INBOUND ORDER
(2CC)	716	ADDRESS	4	XTWREPPA	ADDRESS OF REPLY PARAMETERS
(2D0)	720	CHARACTER	8	XTWREPA (0)	REPLY BUFFER FOR SENDR MACRO
(2D0)	720	ADDRESS	4	XTWRADR	ADDRESS OF REPLY AREA
(2D4)	724	ADDRESS	4	XTWRLNG	LENGTH OF REPLY AREA
(2D8)	728	ADDRESS	4		NOT USED

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(2DC)	732	ADDRESS	4	XTWSMSGA	MESSAGE TO SEND FOR DST TASK
(2E0)	736	ADDRESS	4	XTWSMSGR	REASON CODE FOR MESSAGE
(2E4)	740	ADDRESS	4	XTWSMSGD	DESTINATION FOR MESSAGE
(2E8)	744	ADDRESS	4	XTWSOSPL	ADDRESS OF OLD VERSION SPL
(2EC)	748	ADDRESS	4	XTWSTFWA	TRACE FACILITY WORK AREA
(2F0)	752	ADDRESS	4		NOT USED
(2F4)	756	ADDRESS	4	XTWSFSV (14)	SAVE AREA FOR TCB, REGD, REGE - 9
	WORKAREA FOR ALL IPW$$XT- MODULES				
(32C)	812	CHARACTER	16	XTWASD	SECTION DESCRIPTOR
(33C)	828	BITSTRING	1	XTWACT	ACTION BYTE
		1...		XTWAREP	"X'80'" .. REPLY TO SEND
		.1..		XTWASPL	"X'40'" .. SPL TO PROCESS
		..1.		XTWARST	"X'20'" .. PERFORM RESTART
		...1		XTWASOA	"X'10'" .. SOA-SITUATION TO PROCESS
(33D)	829	BITSTRING	1	XTWSTAT1	STATUS BYTE
		1...		XTWST1CL	"X'80'" .. LST-QUEUE ENTRY FOR D-CMD
		.1..		XTWST1DN	"X'40'" .. NOTHING TO DISPLAY
		..1.		XTWST1MP	"X'20'" .. MESSAGE PROCESSING
		...1		XTWST1RP	"X'10'" .. SPL REQUEST PROCESSING
		.... 1...		XTWST1PM	"X'08'" .. 1ST TIME FOR PUT MESSAGE
		.... .1..		XTWST1MR	"X'04'" .. RESTART FOR MESSAGE PROCESSING
		.... ..1.		XTWST1MS	"X'02'" .. SEND REPLY FOR MSG PROCESSING
		1		XTWST1SD	"X'01'" .. SHORT ON DASD OCCURRED DURING CTL
(33E)	830	BITSTRING	1	XTWSTAT2	STATUS BYTE
		1...		XTWST2NR	"X'80'" .. NEW RECORD TO RETRIEVE
		.1..		XTWST2TR	"X'40'" .. TRACE WANTED
		..1.		XTWST2CT	"X'20'" .. LOGICAL I/F CLOSED BY TR
(33F)	831	BITSTRING	1	XTWTRACF	TRACE FUNCTION
		1		XTWTRRCV	"X'01'" .. TRACE XPCC RECEIVE
		.... ..1.		XTWTRREP	"X'02'" .. TRACE XPCC REPLY
(340)	832	ADDRESS	4	XTWSSPL	SPL SAVED DURING PUT REQUEST
(344)	836		8	XTWINT1	USED BY $XTC FOR CVD @D61QDTR
(34C)	844	ADDRESS	4	XTWPUTAR	ADDRESS OF TEMPORARY WORKAREA
(350)	848	ADDRESS	4	XTWLOPL (3)	PARAMETER LIST FOR IPW$$LO A(0) ADDRESS OF JOB HEADER WORKAREA A(0) ADDRESS OF DATASET HEADER WORKAREA A(0) ADDRESS OF JOB TRAILER WORKAREA
(35C)	860	ADDRESS	4	XTWFMSG	ADDRESS OF FIRST MESSAGE
(360)	864	ADDRESS	4	XTWLMSG	ADDRESS OF LAST MESSAGE
(364)	868	ADDRESS	4	XTWTEMPA	TEMPORARY ADDRESS
(368)	872	SIGNED	4	XTWTEMPL	TEMPORARY LENGTH
(36C)	876	BITSTRING	8	XTWBUF (0)	BUFFER-VALUES
(36C)	876	BITSTRING	1	XTWBUFI	USAGE OF BUFFER
		1...		XTWBUF1	"X'80'" .. USE 1 BUFFER ONLY
(36D)	877	ADDRESS	3	XTWBUFAD	ADDRESS OF BUFFER
(370)	880	SIGNED	4	XTWBUFLN	LENGTH OF BUFFER
(374)	884	SIGNED	4	XTWBUFUS	USED BYTES IN BUFFER
(378)	888	SIGNED	4	XTWBUFWT	LENGTH OF BUFFER TO BE RETRIEVED
(37C)	892	SIGNED	4	XTWBUFGT	LENGTH OF BUFFER USED BY IPW$$XTG
(380)	896	CHARACTER	8	XTWEOJ	BUFFER FOR DOS OR PWR EOJ
(388)	904	SIGNED	4	XTWACIE	ADDR. OF ACIE USED BY $XTM
(38C)	908	CHARACTER	71	XTWACCNT (0)	INTERFACE ACCOUNT RECORD
(38C)	908	CHARACTER	8	XTWACDT	DATE IN SYSGEN FORM
(394)	916	SIGNED	4	XTWACST	CONNECTION START TIME
(398)	920	SIGNED	4	XTWACSP	CONNECTION STOP TIME
(39C)	924	CHARACTER	8	XTWACAP	APPLICATION ID
(3A4)	932	SIGNED	4	XTWACMSG	NUMBER OF PROCESSED MSGES
(3A8)	936	SIGNED	4	XTWACCTL	NUMBER OF PROCESSED CTL'S

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(3AC)	940	BITSTRING	1	XTWACTC	TERMINATION CODE
		.1..		XTWACTSE	"X'40'" .. SEVERE SYSTEM OR POWER ERROR
		..1.		XTWACTKL	"X'20'" .. TERMINATION DUE TO PSTOP KILL
		...1		XTWACTUE	"X'10'" .. SEVERE USER ERROR
		.... 1...		XTWACTAT	"X'08'" .. ABNORMAL USER TERMINATION
		.... .1..		XTWACTPP	"X'04'" .. TERMINATION DUE TO PSTOP
		.... ..1.		XTWACTPD	"X'02'" .. TERMINATION DUE TO PEND
		1		XTWACTOK	"X'01'" .. NORMAL USER TERMINATION
(3AD)	941	BITSTRING	1	XTWACF1	XTW ACCOUNT FLAG BYTE 1
		1...		XTWCE20	"X'80'" .. XWTACDT IS 20YY CENTURY
(3AE)	942	CHARACTER	8	XTWACDEV	DEVICE NAME, IF DST TASK
(3B6)	950	CHARACTER	1	XTWACID	IDENTIFIER = C
		..1. 1.11		XTWACLN	"*-XTWACCNT" LENGTH OF INTERFACE ACCOUNT RECORD
(3B7)	951	BITSTRING	3	XTWARCFB (0)	RETURN AND FEEDBACK CODES
(3B7)	951	BITSTRING	1	XTWARC	RETURN CODE
(3B8)	952	BITSTRING	1	XTWAFB	FEEDBACK CODE
(3B9)	953	BITSTRING	1	XTWAF2	FEEDBACK CODE 2
(3BA)	954	BITSTRING	2		NOT USED
		EXPRESSION		XTWLN	"*-XTWDS" LENGTH OF CONTROL BLOCK
		EXPRESSION		XTWPCAR	"*" CARRIER FOR IPW$SSJ MACRO ASM H V 02 09.44

Spool Environment Block

Definition Macro: IPW$DSP SPB=YES

This control block contains the current printer setup (SETPRT parameter list) and the TRC value. The control block is built by Data Management (IPW$$PD) and will be released by Data Management when the Job Trailer Record of the current spooled entry is successfully written to disk. Each spooled record is examined if it causes a change of the current environment. If so, the spool environment block is updated accordingly. The format is as follows:

Bytes Hex.	Label of Field	Description/Function
00	SPBDS	Start of DSECT
00-0B	SPBSD	Storage descriptor of SPB
0C-0F	SPBATCB	Address of owning task TCB
10	SPBTRC	Current TRC value
11	SPBFLAG	Flag byte
12-13		Reserved for future use
14-57	SPBSETP	SETPRT parameter list
58-5F		Reserved for future use

Spool Environment Header (SEH)

Definition Macro: IPW$DSP SEH=YES

This record is only present as first record in the first DBLK of a DBLK group and is used to provide a backward chain within the DBLK groups that belong to a certain queue entry. The record is built by Data Management (IPW$$PD) when the first DBLK within a DBLK group is allocated. Besides the relative number of the first DBLK of the previous DBLK group it contains an environment section where the various account values (page/record/line count) and the current printer setup is held, as accumulated for the previous DBLK group. The format is as follows:

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
					SPOOL ENVIRONMENT HEADER (SEH)
(0)	0	STRUCTURE	0	SEHDS	LAYOUT OF SPOOL ENVIRONM. HEADER = HEADER OF FIRST DBLK IN DBLKGP
(0)	0	ADDRESS	2	SEHLEN	LENGTH OF SEH RECORD
(2)	2	BITSTRING	2		RESERVED
(4)	4	CHARACTER	4	SEHSD	STORAGE IDENTIFIER, EYECATCHER
(8)	8	ADDRESS	4	SEHDBLK	DBLK NUMBER OF 1ST DBLK OF PREV. DBLK GROUP, BACKWARD CHAIN PTR., RANGING FROM 0 TO N --> 'F..F' IN FIRST DBLK OF A Q-ENTRY --> 'F..F' IN FREE DBLKGP SUBCHAIN RIGHT AFTER COLD START --> UNDEF. IN FREE DBLKGP SUBCHAIN AFTER BEEN USED ONCE --> '0..0' FOR ALL DBLK'S OF QCA !!
(C)	12	SIGNED	2	SEHDBSZ	DBLK SIZE
(E)	14	SIGNED	2	SEHDBGP	DBLK GROUP SIZE
(10)	16	ADDRESS	4	SEHOWNE	OWNER OF DBLK GROUP, MAY BE ... --> 'F..F' - IN FREE DBLKGP SUBCHAIN AFTER D-FILE FORMATTING --> '?..?' - IN FREE DBLKGP SUBCHAIN AFTER BEEN USED ONCE --> '0..0' - OWNED BY QCA (SLOT MGR) --> 'N..N' - OWNED BY Q-REC-# 'NNNN'
(14)	20	BITSTRING	8	SEHSTCK	STORE CLOCK VALUE
(1C)	28	BITSTRING	1	SEHSTFG	STORE CLOCK STATE FLAG
		11.. .11.		SEHFGF	"C'F'" .. STCK SET BY D-FILE FORMAT .. STCK NOT SET BY $$Q1 ALLOC GP.
		11.1 .111		SEHFGP	"C'P'" .. STCK SET BY $$PD PUT DATA
		11.1 1...		SEHFGQ	"C'Q'" .. STCK SET BY $$SQ QCA SLOT MGR. .. STCK NOT SET BY $$Q1 DE-ALL-GP
(1D)	29	BITSTRING	3		RESERVED
		..1.		SEHSLN	"*-SEHDS" LENGTH OF SKELETON SEH-RECORD ALL X=FIELDS: SPOOLING INFO BEING --> 0 - IN FREE DBLKGP SUBCHAIN RIGHT AFTER COLD START --> UNDEF. IN FREE DBLKGP SUBCHAIN AFTER BEEN USED ONCE --> 0 - IN 1ST DBLKGP OF A Q-ENTRY --> VALID IN GROUPS 2-M OF A Q-ENTRY WHICH OWNS M DBLK GROUPS (INFO COLLECTED UP TO ENTRY OF CURRENT DBLKGP, EQUAL TO EXIT OF PREVIOUS GROUP) --> NOT PRESENT FOR ALL QCA DBLK'S
(20)	32	ADDRESS	4	SEHPAGE	X=CURRENT PAGE NUMBER
(24)	36	ADDRESS	4	SEHLINE	X=CURRENT LINE NUMBER
(28)	40	ADDRESS	4	SEHRECD	X=CURRENT RECORD NUMBER
(2C)	44	ADDRESS	4	SEHGPNO	DBLK GROUP NUMBER WITHIN Q-ENTRY --> 0 - IN FREE DBLKGP SUBCHAIN RIGHT AFTER COLD START --> UNDEF. - IN FREE DBLKGP SUBCHAIN AFTER BEEN USED ONCE --> 1-M - IF GROUP BELONGS TO Q-ENTRY WHICH OWNS M DBLK GROUPS
(30)	48	BITSTRING	1	SEHTRC	X=TRC VALUE
(31)	49	BITSTRING	1	SEHFLAG	X=FLAG BYTE
		1...		SEHFHCPY	"X'80'" .. HORIZONTAL COPY ON 4 FLAGS FOR PAGE COUNTING STATE:

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
		.... 1...		SEHFLM	"X'08'" .. LINE MODE (DEFAULT)
		.... .1..		SEHFLMI	"X'04'" .. LINE MODE IDM/IMM RECEIVED
		.... ..1.		SEHFPM	"X'02'" .. PAGE MODE
		1		SEHFPM8	"X'01'" .. PAGE MODE '8B' RECEIVED
(32)	50	BITSTRING	2		UNUSED
(34)	52	BITSTRING	68	SEHSETP	X=SETPRT PARAMETER LIST
(78)	120	BITSTRING	8		RESERVED
(80)	128	BITSTRING	64		RESERVED
		11..		SEHLN	"*-SEHDS" LENGTH OF RECORD

Spool Environment Record (SER)

Definition Macro: IPW$DSP SER=YES

This record is only present as first record in the last DBLK of a DBLK group and is used to chain the DBLK groups together. The record is built by Data Management (IPW$$PD) when the last DBLK within a DBLK group is allocated. Besides the relative number of the first DBLK in the next DBLK group it contains an environment section where the various account values (page/record/line count) and the current printer setup is held. The format is as follows:

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
					SPOOL ENVIRONMENT RECORD (SER)
(0)	0	STRUCTURE	0	SERDS	LAYOUT OF SPOOL ENVIRONMENT RECORD ... = HEADER OF LAST DBLK IN DBLKGP
(0)	0	ADDRESS	2	SERLEN	LENGTH OF SER RECORD
(2)	2	BITSTRING	2		RESERVED
(4)	4	CHARACTER	4	SERSD	STORAGE IDENTIFIER, EYECATCHER
(8)	8	ADDRESS	4	SERDBLK	DBLK NUMBER OF 1ST DBLK OF NEXT.. DBLK GROUP, FORWARD CHAIN PTR., RANGING FROM 0 TO N --> 'F..F' IN LAST DBLK OF A Q-ENTRY --> 'F..F' IN LAST DBLK-FREE-SUBCHAIN --> '0..0' FOR ALL DBLK'S OF QCA !!
(C)	12	SIGNED	2	SERDBSZ	DBLK SIZE
(E)	14	SIGNED	2	SERDBGP	DBLK GROUP SIZE
(10)	16	ADDRESS	4	SEROWNE	OWNER OF DBLK GROUP, MAY BE ... --> 'F..F' - IN FREE DBLKGP SUBCHAIN AFTER D-FILE FORMATTING --> 'F..F' - IN FREE DBLKGP SUBCHAIN WHEN DBLKGP TRACING ON --> '?..?' - IN FREE DBLKGP SUBCHAIN WHEN DBLKGP TRACING OFF --> '0..0' - OWNED BY QCA (SLOT MGR) --> 'N..N' - OWNED BY Q-REC-# 'N..N'
(14)	20	BITSTRING	8	SERSTCK	STORE CLOCK VALUE
(1C)	28	BITSTRING	1	SERSTFG	STORE CLOCK STATE FLAG
		11.. .11.		SERFGF	"C'F'" .. STCK SET BY D-FILE FORMAT
		11.. ...1		SERFGA	"C'A'" .. STCK SET BY $$Q1 ALLOC GROUP
		11.1 .111		SERFGP	"C'P'" .. STCK SET BY $$PD PUT DATA
		11.1 1...		SERFGQ	"C'Q'" .. STCK SET BY $$SQ QCA SLOT MGR.
		11.. .1..		SERFGD	"C'D'" .. STCK SET BY $$Q1 DE-ALLOC GRP.
(1D)	29	BITSTRING	3		RESERVED
		..1.		SERSLN	"*-SERDS" LENGTH OF SKELETON SER-RECORD ALL X=FIELDS: SPOOLING INFO BEING --> 0 - IN FREE DBLKGP SUBCHAIN --> VALID IN GROUP 1-(M-1) OF Q-ENTRY WHICH OWNS M DBLK GROUPS (INFO COLLECTED UP TO END OF CURRENT DBLKGP) --> 0 - IN GROUP M(LAST) OF Q-ENTRY --> NOT PRESENT FOR ALL QCA DBLK'S
(20)	32	ADDRESS	4	SERPAGE	X=CURRENT PAGE NUMBER
(24)	36	ADDRESS	4	SERLINE	X=CURRENT LINE NUMBER
(28)	40	ADDRESS	4	SERRECD	X=CURRENT RECORD NUMBER
(2C)	44	ADDRESS	4	SERGPNO	DBLK GROUP NUMBER WITHIN Q-ENTRY --> 0 - IN FREE DBLKGP SUBCHAIN --> 1-M - IF GROUP BELONGS TO Q-ENTRY WHICH OWNS M DBLK GROUPS, AND SPOOLED DATA IN DBLK 0 - BUT NO SPOOLED DATA IN DBLK
(30)	48	BITSTRING	1	SERTRC	X=TRC VALUE
(31)	49	BITSTRING	1	SERFLAG	X=FLAG BYTE
		1...		SERFHCPY	"X'80'" .. HORIZONTAL COPY ON 4 FLAGS FOR PAGE COUNTING STATE:

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
		.... 1...		SERFLM	"X'08'" .. LINE MODE (DEFAULT)
		.... .1..		SERFLMI	"X'04'" .. LINE MODE IDM/IMM RECEIVED
		.... ..1.		SERFPM	"X'02'" .. PAGE MODE
		1		SERFPM8	"X'01'" .. PAGE MODE '8B' RECEIVED
(32)	50	BITSTRING	2		UNUSED
(34)	52	BITSTRING	68	SERSETP	X=SETPRT PARAMETER LIST
(78)	120	BITSTRING	8		RESERVED
(80)	128	BITSTRING	64		RESERVED
		11..		SERLN	"*-SERDS" LENGTH OF RECORD

Storage Control Block (SCB)

Definition Macro: IPW$DSC

The storage control block is used to control access to the storage management routines and to allocate storage pages as required by the routines. The format of the storage control block is as follows:

Offset Hex	Type	Len	Name (Dim)	Description
(0)	CHARACTER	16	SCSD	SECTION DESCRIPTOR
(10)	ADDRESS	4	SCFP	FIRST FIXED PAGE
(14)	ADDRESS	4	SCFBCW	START OF FIRST BUFFER
(18)	BITSTRING	4	SCEB	EVENT CONTROL BLOCK
(1C)	SIGNED	4	SCLK	LOCK WORD
	SHIFT VALUE AND PAGE SIZE ARE COPIED DURING INITIALIZATION FROM THE PAGE MANAGEMENT COMAREA OF THE SYSTEM INTO THE FOLLOWING FIELDS:			
(20)	SIGNED	2	SCADPN	SHIFT AMOUNT ADDR TO PAGE#
(22)	SIGNED	2	SCCUSH	REAL STORAGE CUSHION SIZE
(24)	SIGNED	4	SCPGSIZE	PAGE SIZE IN BYTES
	SINCE THE STORAGE MANAGEMENT ROUTINES ARE USED TO PROVIDE REGISTER SAVE AREAS FOR TASK USE THE STORAGE CONTROL BLOCK MUST ITSELF CONTAIN A REGISTER SAVE AREA FOR USE BY THE STORAGE MANAGEMENT ROUTINES.			
(28)	CHARACTER	40	SCTR (0)	REGISTER SAVE AREA
(28)	SIGNED	4	SCRE	TASK REGISTER 14
(2C)	SIGNED	4	SCRF	TASK REGISTER 15
(30)	SIGNED	4	SCR0	TASK REGISTER 0
(34)	SIGNED	4	SCR1	TASK REGISTER 1
(38)	SIGNED	4	SCR2	TASK REGISTER 2
(3C)	SIGNED	4	SCR3	TASK REGISTER 3
(40)	SIGNED	4	SCR4	TASK REGISTER 4
(44)	SIGNED	4	SCR5	TASK REGISTER 5
(48)	SIGNED	4	SCR6	TASK REGISTER 6
(4C)	SIGNED	4	SCR7	TASK REGISTER 7
	THE STORAGE MANAGEMENT PAGE CONTROL TABLE ACTS AS AN ISOMORPHIC MAP OF THE FIXABLE AREA WITHIN THE POWER ADDRESS SPACE IN WHICH EACH PAGE CONTROL BIT REPRESENTS A SINGLE PAGE OF ADDRESS SPACE. BIT POSITIONS WITH VALUE 1 REPRESENT PAGES WHICH ARE FIXED IN REAL STORAGE VIA PFIX MACRO. BIT POSITIONS WITH VALUE 0 REPRESENT PAGES WHICH ARE NOT YET FIXED OR WHICH ARE EXPLICITLY FREED VIA PFREE MACRO. THE PAGE CONTROL TABLE IS DEFINED WITH ALL PAGES NOT FIXED AND IS PROPERLY INITIALIZED BY THE POWER START-UP ROUTINES WHICH FIX THE FIRST AND LAST PAGE OF STORAGE AVAILABLE TO THE VSE/POWER PARTITION AT THAT TIME AND WHICH INSERT A FIRST AND LAST BUFFER CONTROL WORD (BCW) INTO THESE PAGES.			
(50)	BITSTRING	64	SCBT	PAGE CONTROL TABLE
	WORKAREA REQUIRED BY THE STORAGE MANAGEMENT ROUTINES DURING PFIX/PFREE REQUESTS.			
(90)	CHARACTER	12	SCPF (0)	PAGE FIX/FREE WORK AREA
(90)	SIGNED	4		PAGE VIRTUAL ADDRESS
(94)	SIGNED	4		PAGE LENGTH (-1)
(98)	BITSTRING	4		END OF LIST INDICATOR
(9C)	SIGNED	4	SCCUR	CURRENT # OF BYTES $RSW'D(+)
(A0)	BITSTRING	1	SCFLG	FLAG BYTE
	1...		SCNBDY	"X'80'" ..DON'T CROSS PAGE BOUNDARY

Offset Hex	Type	Len	Name (Dim)	Description
(A1)	BITSTRING	1	SCCOBY	COPY OF PAGE-BIT BYTE
(A2)	BITSTRING	2		UNUSED
	1.1. .1..		SCLN	"*-SCDS" LENGTH OF CONTROL BLOCK

Notes:

1. Since the storage management routines are used to provide register save areas for task use, the storage control block must contain a register save area for use by the storage management routines.
2. The storage assignment table is like a map of the fixable area within the VSE/POWER address space in which each bit represents a single page of address space. The bit is on, if the page is fixed.

   The storage assignment table is defined with all pages free and is properly initialized by the VSE/POWER startup routines to reflect the amount of real storage available to the VSE/POWER partition at that time.
3. Three fullwords used as a work area by the page-fix and page-free routines. The first word is used to contain the address of the first byte of the page to be fixed or freed; the second word contains binary 2047 (page size minus one); and the third word contains X'FF' in its high-order byte to act as a list terminator.

How to Locate: Refer to Figure 152 on page 731 in Chapter 6, “Diagnostic Aids.”

System Information Area (SIA)

Definition Macro: IPW$DEF PSYS=YES

The system information area is used by a user program to access VSE/POWER generation and configuration information.

Offset Hex	Type	Len	Name (Dim)	Description
(0)	CHARACTER	8	PSYSNODE	VSE/POWER PNET NODE NAME
(8)	CHARACTER	1	PSYSSID	VSE/POWER SHARED SYSID 1-9 X'00' - VSE/POWER IS DOWN X'40' - VSE/POWER NON-SHARED IS UP C'1'-C'9' - VSE/POWER SHARED SYSID IS UP
(9)	BITSTRING	1	PSYSFLG1	VSE/POWER SYSTEM INFO FLAG BYTE
	1...		PSF1SKP	"X'80'" .. 'SET SKIP=YES' ACTIVE
(A)	CHARACTER	6		UNUSED

How to Locate: The access to the SIA is granted by either the GETFLD FIELD=POWSYS macro or the SYSCOM.IJBPSYSI byte of the VSE system communication area.

Tape Control Block (TBB)

Definition Macro: IPW$DTB

This control block dynamically created to satisfy requirements of VSE/POWER tasks utilizing tape as intermediate storage. Its format as it is printed in a dump is as follows:

The IPW$DTB macro is issued by VSE/POWER phases IPW$$OF, IPW$$OT and IPW$$SY.

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(0)	0	STRUCTURE		TBDS	, DEFINE DUMMY SECTION
(0)	0	CHARACTER	4	TBSD	STORAGE DESCRIPTOR
(4)	4	SIGNED	4	TBPB	PHYSICAL UNIT BLOCK ADDRESS
(8)	8	BITSTRING	1	TBFU	FUNCTION CONTROL BYTE
		1...		FBTB	"X'80'" .. BUILD TBB REQUEST
		.1..		FOPT	"X'40'" .. OPEN TAPE REQUEST
		..1.		FCLT	"X'20'" .. CLOSE TAPE RQUEST
		...1		FCON	"X'10'" .. CONTINUATION REQUESTED
		.... 1...		FEOVC	"X'08'" .. BAM EOV,CONTINUED QREC
		.... .1..		FEOVN	"X'04'" .. BAM EOV,NON-CONT. QREC
		.... ..1.		FINP	"X'02'" .. INPUT PROCESSING
		1		FOUT	"X'01'" .. OUTPUT PROCESSING
					FBTB +FCLT = DELETE TBB REQUEST
					FEOVC+FCLT = MOUNT 1ST VOL REQUEST
					FEOVC+FOPT = MOUNT LAST VOL REQUEST
					FEOVN+FCLT = SYSIN FEOV REQUEST
(9)	9	BITSTRING	1	TBFG	TBB FLAG BYTE 1
		1...		TEOF	"X'80'" .. EOF INDICATED
		.1..		TEOV	"X'40'" .. EOV INDICATED
		..1.		TDMD	"X'20'" .. DATA MODE,NOT LABEL OPERATION
		...1		TBLK	"X'10'" .. BLOCKED DATA
		.... 1...		TUNL	"X'08'" .. UNLABELLED TAPE
		.... .1..		TMVF	"X'04'" .. MULTI-VOLUME-FILE
		.... ..1.		TMFI	"X'02'" .. MULTI-FILE-VOLUME
		1		TTWA	"X'01'" .. TEMPORARY WORKAREA AVAILABLE
(A)	10	BITSTRING	1	TBFG2	TBB FLAG BYTE 2
		1...		TBFND	"X'80'" .. NON DISP.QUEUE PROCESS'G
		.1..		TBSFN	"X'40'" .. SELECT ENTRY FOUND
		..1.		TBSAL	"X'20'" .. SELECT ALL QUEUE ENTRIES
		...1		TBFTS	"X'10'" .. TAPE SPOOLING READ ENTRY
		.... 1...		TB1QS	"X'08'" .. AT LEAST 1 QSET ON LAB TP
		.... .1..		TBPDMB	"X'04'" .. PICKUP ALREADY OWNS DMB
					X'02' .. UNUSED
					X'01' .. UNUSED
(B)	11	BITSTRING	1	TBFG3	TBB FLAG BYTE 3
		1...		TBCARTE	X'80' .. CARTRIDGE LABEL TAPE EDF
		.1..		TBSKFSF	X'40'.. SKIP FSF(CACHING TP UNIT)
		.1..		TB3490F	X'40'.. (No longer used)
		..1.		TBIGNDR	X'20' .. PGO IGNORE REPLY
		...1		TBRCEMP	X'10' .. 1Q2BI QUEUES/TAPE EMPTY
		.... 1...		TBRCPNF	X'08' .. 1Q2BI PICKUP NOTHING FOUND
(C)	12	BITSTRING	1	TBSM	SPECIFIED MODE SETTING ... ... REFER ALSO TO TCF8MS
(D)	13	CHARACTER	1	TBDT	DEVICE IDENTIFIER
(E)	14	ADDRESS	3	TBCU	PHYSICAL UNIT NUMBER (CUU)
	CHANNEL PROGRAM (CCB & CCW'S)				
(18)	24	DBL WORD	8	(0)	FORCE DOUBLE-WORD ALIGNMENT
(18)	24	CHARACTER	16	TBCB (0)	COMMAND CONTROL BLOCK

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(18)	24	BITSTRING	2	TBRS	RESIDUAL COUNT
(1A)	26	BITSTRING	2	TBCM	COMMUNICATION BYTES
		1..1 111.		TCOM	"B'10011110'" ..CCB COMM. BYTE (X'9E')
(1C)	28	BITSTRING	1	TBCS	CHANNEL AND DEVICE STATUS
(1D)	29	BITSTRING	1	TBC1	CHANNEL AND DEVICE STATUS
(1E)	30	BITSTRING	2	TBLU	LUB INDEX
(20)	32	ADDRESS	4	TBCA	CCW ADDRESS
(24)	36	ADDRESS	4		CCW ADDRESS IN CSW
(28)	40	DBL WORD	8	TBCW (0)	CHANNEL COMMAND WORD
(28)	40	BITSTRING	1	TBCC	WRITE COMMAND CODE
(29)	41	ADDRESS	3	TBRA	DATA ADDRESS
(2C)	44	BITSTRING	2	TBWS	FLAGS
(2E)	46	SIGNED	2	TBCT	COUNT
		.... 1...		TLNC	"*-TBCW" CCW-LENGTH
MULTI-PURPOSE WORKAREA					
(30)	48	CHARACTER	17	TBLA	LABEL............ FILE LABEL SAVE AREA
THE FOLLOWING AREA IS USED DURING OPERATOR INTERVENTION					
(41)	65	CHARACTER	13	TBWA (0)	TBB WORKAREA
(41)	65	CHARACTER	1	TBWR	LENGTH OF REPLY AREA
(42)	66	CHARACTER	12	TBW1	REPLY AREA
VARIOUS OTHER CONTROL FIELDS					
(4E)	78	BITSTRING	1	TBRC	REASON CODE
(4F)	79	BITSTRING	1		NOT USED
(50)	80	SIGNED	2	TBCN	NUMBER OF CCW IN I/O-AREA
(52)	82	BITSTRING	2		NOT USED
(54)	84	SIGNED	4	TBAR	REAL ADDRESS OF I/O AREA
(58)	88	SIGNED	4	TBAV	VIRTUAL ADDRESS OF I/O AREA
(5C)	92	SIGNED	4	TBIO	ADDRESS OF CCW IN TBB
(60)	96	SIGNED	2	TBDL	LENGTH OF DATA REC (SYSIN)
(62)	98	SIGNED	2	TBRL	LOGICAL RECORD LENGTH
(64)	100	BITSTRING	1	TBBF	BLOCK FACTOR
(65)	101	BITSTRING	1	TBQI	USED TO SAVE Q-ID FOR OFFLOAD
(66)	102	BITSTRING	1	TBSS	SENSE BYTE 1
(67)	103	BITSTRING	1	TBSI	INDICATION OF SUCCESS
		....		TTSI	"X'00'" .. NOTHING TO SAVE INDICATION
		.1..		TB40	"X'40'" .. NORMAL - CONTINUE EXECUTION
(68)	104	BITSTRING	1	TBSQF	POFFLOAD SEARCH QUEUE FOUND
(69)	105	BITSTRING	1	(3)	NOT USED
BAM PROCESSING ADDITIONS					
(6C)	108	SIGNED	4	TBLNKR7	LINKAGE REGISTER SAVE AREA
(70)	112	SIGNED	4	TBLNKR9	LINKAGE REGISTER SAVE AREA
(74)	116	SIGNED	4	TBLNK1	LINKAGE REG SAVEAREA 1
(78)	120	SIGNED	4	TBDTFWA	DTFMT WORKAREA
(7C)	124	SIGNED	4	TBDTFMT	DTFMT POINTER
(80)	128	SIGNED	4	TBDTFMTC	DTFMT COPY
(84)	132	SIGNED	4	TBDTFMTL	DTFMT LENGTH
(88)	136	BITSTRING	1	TB1Q5A	MSG 1Q5AI RETURN CODE
(89)	137	BITSTRING	1	TBTEMP	TEMP SAVE AREA
(8A)	138	BITSTRING	1	TBTMRC	TEMP RETURN CODE
(8B)	139	BITSTRING	1	TBSUBRC	SUBROUTINE RETURN CODE QUEUE RECORD BEING PROCESSED
(8C)	140	CHARACTER	8	TBQRDY	.. DATE
(94)	148	CHARACTER	35	TBQRSA	.. TIMES,USERINFO, ETC.
(B7)	183	BITSTRING	1	TBQRSN	.. SUFFIX

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(B8)	184	BITSTRING	1		RESERVED
(B9)	185	BITSTRING	7	TBBAMLB	BAM LABEL FILENAME
(C0)	192	SIGNED	2	TBDBLK	LENGTH OF DBLK
(C2)	194	SIGNED	2	TBQREC	LENGTH OF QREC
(C4)	196	BITSTRING	1	TB1QG01	MESSAGE 1QGOA FIELD 1
(C5)	197	BITSTRING	1	TB1QG02	MESSAGE 1QGOA FIELD 2
(C6)	198	SIGNED	2		UNUSED
(C8)	200	SIGNED	4	TBAVQREC	TEMP QREC WORKAREA
(CC)	204	SIGNED	4	TBRASAVE	SAVE AREA FOR TBRA
(D0)	208	SIGNED	4		UNUSED
		11.1 .1..		TBLN	"*-TBDS" LENGTH OF CONTROL BLOCK
	TAPE SPECIFIC INDICATORS				
		1		TUEX	"X'01'" UNIT EXCEPTION (EOV/EOF)
		...1		TSDT	"X'10'" 'SUPPRESS DATA TRANSFER'
		..1.		TSLI	"X'20'" SLI SET ON IN CCW
		.1..		CCBSLI	"X'40'" INCORRECT LENGTH IND. IN CCB
		1.11 1111		TBER	"X'FF'-CCBSLI" CATASTROPHICAL TAPE ERROR

How to Locate: Refer to Figure 152 on page 731 in Chapter 6, “Diagnostic Aids.”

Task Control Block (TCB)

Definition Macro: IPW$DTC

Each VSE/POWER task is equipped with a task control block which is created in fixed storage and is used to establish the identity of the task and to preserve its status when it is not in active control of the central processor.

The TCB is divided into the following main areas:

- Task state
- Task management fields
- Task register save area
- Linkage register save area
- General task work area and Task extensions fields

When the TCB belongs to a command processor task, the general task work area is replaced by command processor control fields. Refer to the "Command Processor Control Block" paragraph in this chapter.

Note: The first characters of the labels in the control block vary according to the generated DSECT or declaration (PL/S).

TC Current TCB

IT Initiator/terminator TCB

OC Operator command processor

TN Used to address a TCB other than the task's own TCB. (To enable a task to address the TCB of another task.)

TP Used to address a TCB other than the task's own TCB. (To enable a task to address the TCB of another task.)

TCB Used to address a TCB other than the task's own TCB in the PL/S listings.

TCB State

At any time, each task within the VSE/POWER must be in one or another of a set of task states. The state of each task is defined by the single alphameric character in byte 28 of the associated task control block, and this in turn determines what action the task management routines must take when the task is examined for dispatch. Task states are normally set by the task itself whenever one of the task management macros is issued. The task management routines, the command processing task and the execution reader tasks are privileged, however, in that they may modify the task state of tasks other than themselves.

Note: Task states can also be set by the page fault appendage routine.

Task States	Char.	Task Condition	Routine
Not dispatchable	I	Task is inactive	TM10
	P	Page fault in process	TM10
	O	Waiting for operator response	TM10
Conditionally dispatchable	L	Waiting for locked resource	TM30
	X	Wait for mixed ECB and class anchors	TM40
	M	Wait on multiple CCB or ECB posting	TM50
	Q	As for M state, except event may never occur	
	C	Wait on single CCB or ECB posting	
	E	Wait on single ECB posting	
	S	As for C state, except event may never occur	
	B	Wait on RJE,BSC or networking event	TMB0
Immediately dispatchable	D	Dispatch task immediately	TM90
Running	R	Task is running	N/A
Partition wait	W	Waiting for dispatch from supervisor	TM20

TCB Task Register Save Area (TRSA)

The fields in this area in a TCB record the contents of registers 12 through 9 whenever entry is made to task selection. If the task state is set to R (running) the values in the fields record the contents of the registers when the task was most recently given control. If the task state is set to any other value the fields contain the current contents of the registers associated with the task. The format of a TCB is as follows:

Label of Field	Description/Function
TCTR	Register 12 - asynchronous address register ('task PSW'). R12 contains the address of the first instruction to be executed when the task is dispatched. The first byte contains the condition code and the program mask bits in the form in which they are loaded by BAL instructions. This is also true when the information is provided by the page fault appendage routine.
TCRD	Register 13 - save area register which may contain the address of either the first (or only) or second linkage register save area depending on the hierarchy level of the caller.
TCRE	Register 14 - linkage register 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.

Label of Field	Description/Function
TCRF	Register 15 - entry point register 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 function to be executed. When not required for this purpose the register is available for general use.
TCRO	Register 0 - parameter and work register is used to pass parameters to and from invoked routines. When not required for this purpose the register is available for general use.
TCR1	Register 1 - parameter and work register may address a control block or control block list on which the task is at present waiting. For a task in C or S state it will point to a conventional VSE CCB for a VSE/POWER ECB. For a task in M or Q state, it will point to an ECB or CCB list.
TCR2	Register 2 - linkage and work register is used by 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.
TCR3	Register 3 - resource address register may contain the address of a resource control block on which the task is at present waiting (task in L state). When not required for this purpose the register is available for general task use.
TCR4	Register 4 - work register
TCR5	Register 5 - work register. If the task owns queue space, this register will address the queue record.
TCR6	Work register (may address the DMB). In an execution processor task, it addresses the partition control block.
TCR7	Work register. In an execution processor task this register addresses the user CCB.
TCR8	Work register. In an execution processor task this register addresses current channel command. In a physical routine, it points to PWS.
TCR9	Base register for highest level of code used by task.

TCB

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
TASK CONTROL BLOCK (TCB)					
TASK MANAGEMENT FIELDS					
					THE FOLLOWING FIELDS DEFINE THE IDENTITY OF THE TASK, ESTABLISH ITS POSITION IN THE TASK LIST, RECORD PAGE FAULTS PENDING, AND DEFINE THE TASK STATE AT ANY POINT IN TIME.
(0)	0	CHARACTER	16	TCSD (0)	STORAGE DESCRIPTOR
(0)	0	CHARACTER	4	TCBI	BLOCK IDENTIFIER
(4)	4	CHARACTER	4	TCTI	TASK IDENTIFIER: C'O CP' - COMMAND PROCESSOR TASK C'I IT' - INITIATOR TASK C'T TT' - TERMINATOR TASK C'T TI' - TIMER TASK C'RRDR' - LOCAL READER TASK C'WLST' - LOCAL PRINTER TASK C'WPUN' - LOCAL PUNCH TASK C'E XX' - EXECUTION PROCESSOR TASK. XX SPECIFIES THE PARTITION REQUESTING THE TASK. C'1'-C'5 ' TCB BELONGS TO RJE TASK IN THIS CASE THREE REMAINING BYTES WILL INDICATE THE TYPE OF TASK. (RDR, LST, PUN, LGN, LGF, OR MSG.) C'LRLM' - LINE MANAGER TASK C'P PS' - PRINT STATUS TASK C' ACT' - ACCOUNT TASK C'J ' - SPOOL MANAGER TASK. THE THREE REMAINING BYTES IND THE TYPE OF TASK.(RDR,LST,OR SPM.) C'LSNA' - SNA TASK C'NTFY' - NOTIFY TASK C'LLDR' - PNET DRIVER C'NRVN' - NETWORK RECEIVER TASK N (N=BLANK FOR CONSOLE TASK) C'NTRN' - NETWORK TRANSMITTER N (N=BLANK FOR CONSOLE TASK) C'NCT ' - PNET SESSION EST'D C'NDT ' - PNET SESSION DISCONNECT C'XHBT' - OCCF HEARTBEAT TASK C'XMAS' - SAS MASTER TASK C'XSAS' - SAS USER TASK C'XDEV' - DEVICE SERVICE TASK C'YTES' - TIME EVENT SCHEDULING C'DPST' - DYNAMIC PART.SCHEDULING
(8)	8	CHARACTER	4	TCCU	PHYSICAL DEVICE IDENTIFIER Physical device address. If byte 0 of the task ID field = '1' - '9', then TCCU contains the RJE line number, or 'SNA' for all RJE,SNA TCBs. 'PSP' (for RDR task) and 'GSP' (for LST task) are used for PUTSPOOL and GETSPOOL, CTLSPOOL and Spool Access Support processing
(C)	12	CHARACTER	4	TCRI (0)	RJE-ID/TAPE CUU/RCV-TSM TYPE
(C)	12	BITSTRING	1	TCFL	BINARY FORMAT

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(D)	13	CHARACTER	3	TCRM	CHARACTER FORMAT Identifies the terminal ID requiring the task. If TCRI = X'00' then task started as result of command invoked by the central operator. For PNET RCV/TSM task following subspecification is presented: 'CON' = console task 'JOB' = job receiver/transmitter 'OUT' = output receiver/transmitter
(10)	16	ADDRESS	4	TCTP	ADDRESS OF PREVIOUS TASK TCB
(14)	20	ADDRESS	4	TCTN	ADDRESS OF NEXT TASK TCB If the present is the last TCB in the chain, the address in TCTN is that of the wait control block (WCB).
(18)	24	SIGNED	4	TCPF	PAGE FAULT REQUEST WORD Contains page fault request information resulting from a page fault interrupt. Contents of R13, passed from VSE/Advanced Functions supervisor and saved for page management in the event of a page fault occurring during execution of the task. The field is set to binary zeros when no page fault request condition is present; hence, it will contain binary zeros during the time that the task is in control of the central processor.
(1C)	28	SIGNED	4	TCSF (0)	TASK SELECTION FIELD
(1C)	28	BITSTRING	1		.. TASK STATE (SEE BELOW)

TASK STATE VALUES

'B' = X'C2' TASK WAITS ON RJE,BSC OR PNET EVENT
'C' = X'C3' TASK WAITS ON SINGLE CCB/ECB POSTING
AND CHECKS UNRECOVERABLE I/O ERROR
'D' = X'C4' TASK DISPATCHABLE
'E' = X'C5' TASK WAITS ON SINGLE ECB POSTING
'I' = X'C9' TASK INACTIVE
'L' = X'D3' TASK WAITING ON LOCKED RESOURCE
'M' = X'D4' TASK WAITING ON MULTIPLE CCB/ECB
POSTING
'O' = X'D6' TASK WAITING ON OPERATOR RESPONSE
'P' = X'D7' TASK PAGE FAULT IN PROCESS
'Q' = X'D8' TASK WAITING ON MULTIPLE CCB/ECB
POSTING, BUT MAY NEVER OCCUR
'R' = X'D9' TASK IS RUNNING (ONLY ONE TASK)
'S' = X'E2' TASK WAITING ON SINGLE CCB/ECB
POSTING, BUT MAY NEVER OCCUR,
AND CHECKS UNRECOVERABLE I/O ERROR

(1D)	29	ADDRESS	3		.. NUCLEUS TASK ROUT. ADDR
(20)	32	SIGNED	4	TCCT (4)	TASK CLASS LIST
		.... .1..		TC#C	"(*-TCCT)/4" .. NUMBER OF CLASS ENTRIES Up to four different classes can be specified simultaneously for any task, except RDR task. For each class identifying character an entry is made in the TCCT field in the TCB for that task. The first byte of each entry contains the class, and the remaining three bytes contain an address of an ECB in the master class table area (in DMB). The task class list is shown in Figure 145 on page 712.
(30)	48	BITSTRING	1		LIST DELIMITER

OVERLAY AREA USED BY X-PARTITION SPOOL MANAGER TASKS

(24)	36	BITSTRING	1		SPOOL MGMT LIST DELIM'TER
(25)	37	ADDRESS	3	TCEWA	ADDR. OF WS FOR EXTRACT
(28)	40	BITSTRING	1	TCIQ	SPOOL MGMT QUEUE ID
(29)	41	BITSTRING	1		UNUSED
(2A)	42	BITSTRING	1	TCSG	SPOOL MG GEN PURPOSE BYTE
		.... ..1.		TC1T	"X'02'" .. 1ST TIME BUFF'ED GETSP X'01' .. PUTSPOOL DASD SOS MSG
(2B)	43	CHARACTER	1	TCSS	SPOOL MANAGEMENT SWITCH

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
		CHARACTER		TCIW	"C'I'" ..LOGICAL WRITER INITIALIZED
		CHARACTER		TCOW	"C'O'" ..OPEN LOGICAL WRITER
		CHARACTER		TCCW	"C'C'" ..CLOSE LOGICAL WRITER
(2C)	44	SIGNED	4	TCER	ADDR(USER X-PART XECB)
	TERMINATION TYPE				
(31)	49	BITSTRING	1	TCTT	TERMINATION TYPE - SEE BELOW
		CHARACTER		TT40	"C' '" ..NORMAL - CONTINUE EXECUTION
		CHARACTER		TTCU	"C'U'" ..UNRECOVERABLE I/O ERROR
		CHARACTER		TTCX	"C'X'" ..TASK CANCEL CONDITION
		CHARACTER		TTCC	"C'C'" ..PCANCEL COMMAND ISSUED
		CHARACTER		TTCF	"C'F'" ..PFLUSH COMMAND ISSUED
		CHARACTER		TTCE	"C'E'" ..STOP AT END OF JOB
		CHARACTER		TTCS	"C'S'" ..STOP IMMEDIATELY
		CHARACTER		TTCH	"C'H'" ..PFLUSH WITH HOLD ISSUED
		CHARACTER		TTCR	"C'R'" ..STOP IMMEDIATELY AND RESTART
		CHARACTER		TTCB	"C'B'" ..STOP WITH 'BAD 9346 ENTRY
		CHARACTER		TTNT	"C'N'" ..'NEWTAP' INDICATION
		CHARACTER		TTIG	"C'I'" ..'IGNORE' INDICATION
		CHARACTER		TTIO	"C'Y'" ..QUEUE/DATA FILE I/O ERROR
		CHARACTER		TTCL	"C'L' ..STOP WITH QUEUE ENTRY DUE TO INVALID RECORD LENGTH.
(32)	50	BITSTRING	1	TCJB	JOB BOUNDARY SWITCH X'FF' ..JOB IN PROCESS
	FUNCTION TRACE INDICATOR				
(33)	51	BITSTRING	1	TCFT	FUNCTION TRACE INDICATOR
	INPUT PROCESSING				
		CHARACTER		FTNQ	"C'N'" ..GET NEXT SET FROM QUEUE
		CHARACTER		FTDQ	"C'D'" ..DELETE SET FROM QUEUE
		CHARACTER		FTFQ	"C'F'" ..FREE QUEUE SET IN PROCESS
		CHARACTER		FTQN	"C'S'" ..GET NEXT QUEUE RECORD
		CHARACTER		FTGD	"C'G'" ..GET DATA RECORD IN PROCESS
		CHARACTER		FTRI	"C'I'" ..SPOOL FILE READY FOR INPUT
	OUTPUT PROCESSING				
		CHARACTER		FTRQ	"C'R'" ..RESERVE QUEUE RECORD
		CHARACTER		FTAQ	"C'A'" ..ADD TO QUEUE IN PROCESS
		CHARACTER		FTPD	"C'P'" ..PUT DATA RECORD IN PROCESS

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
		CHARACTER		FTRO	"C'O'" ..SPOOL FILE READY FOR O/P
	GENERAL EQUATES				
		CHARACTER		FTCH	"C'U'" ..SET DELETED BUT NOT FREED
		CHARACTER		FTES	"C'E'" ..END OF QUEUE SET PROCESS
		CHARACTER		FTPA	"C'L' ..PUT ACCOUNT REC IN PROCESS
		CHARACTER		FTSM	"C'Z'" ..SLOT MANAGER ACTIVE
		....		FT00	"X'00'" ..TCB JUST INITIALIZED
		.1..		FT40	"X'40'" ..PROCESSING COMPLETE
		CHARACTER		FTTT	"C'T'" .. TIMER TASK ACTIVE
		CHARACTER		FTIN	"C'X'" .. INITIALIZATION ACTIVE
		CHARACTER		FTBC	"C'B'" .. COLD START PROCESSING
		CHARACTER		FTTP	"C'Y'" .. TERMINATION PROC ACTIVE
		CHARACTER		FTQM	"C'M'" .. DBLK GROUP ALLOC/DE-ALLOC
		CHARACTER		FTQQ	"C'Q'" .. DBLK GROUP DE-ALLOC-READ
		CHARACTER		FTE1	"C'1'" .. D-FILE EXTENSION I/O
		CHARACTER		FTE2	"C'2'" .. D-FILE EXTENSION I/O FAILED
	TASK ECB AND OTHER CONTROL FLAGS				
(34)	52	SIGNED	4	TCEB (0)	EVENT CONTROL BLOCK
(34)	52	BITSTRING	1	TCDB	DOUBLE BUFFER INDICATOR
		CHARACTER		TCB2	"C'2'" .. DOUBLE BUFFER FLAG C'N' .. DON'T CLEAR DOUBLE BUFFER
(35)	53	BITSTRING	1	TCCB	FUNCTION COMMUNICATION BYTE
		...1		TCCA	"X'10'" .. ANCHOR ADDR SPECIFIED
		..1.		TCVR	"X'20'" .. VTAM REQUEST PENDING
		.1..		TCNR	"X'40'" .. NODAL MESSAGE RECORD
		1...		TCKP	"X'80'" .. LEAVE IN QUEUE
		.... 1...		TCRAS	"X'08'" .. RELEASE ALL VIRT STORAGE
		.... .1..		TCSSM	"X'04'" .. MSG DESTINED FOR SUBSYS
		.... ..1.		TCNP	"X'02'" .. DON'T BUILD STORAGE PRFX
		1		TCAE	"X'01'" .. TASK ACCEPTS I/O ERORS
(36)	54	BITSTRING	1	TCEP	EVENT POST BYTE
		1...		TCEO	"X'80'" EVENT POST BIT ON SETTING
		.1..		TCBSCLV	"X'40'" EVENT BIT BSC-WAIT 'B'
		..1.		TCQRDR	"X'20'" POST BIT FOR QUIESCE RDR I/O
(37)	55	BITSTRING	1	TCSI	SPOOLING INDICATOR
		CHARACTER		TCTSP	"C'T'" .. IF SPOOLING TO TAPE C'N' .. 'NATIVE' TAPE TASK - UNIT ASSGN'DX C'F' .. TASK BRING UP 'FAILED'
(38)	56	ADDRESS	4		UNUSED
(3C)	60	ADDRESS	4		UNUSED

TASK REGISTER SAVE AREA

THE FOLLOWING FIELDS RECORD THE CONTENTS OF THE GENERAL PURPOSE REGISTERS 12 THROUGH 9 WHENEVER ENTRY IS MADE TO TASK SELECTION. IF THE TASK STATE IS SET TO 'R' (RUNNING) THE VALUES IN THE FIELDS RECORD THE CONTENTS OF THE REGISTERS WHEN THE TASK WAS GIVEN CONTROL. IF THE TASK STATE IS SET TO ANY OTHER VALUE THE FIELDS CONTAIN THE ACTUAL CONTENTS OF THE REGISTERS ASSOCIATED WITH THE TASK.

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(40)	64	CHARACTER	56	TCTR (0)	TASK REGISTER SAVE AREA
(40)	64	SIGNED	4	TCRC	TASK REGISTER 12
(44)	68	SIGNED	4	TCRD	TASK REGISTER 13
(48)	72	SIGNED	4	TCRE	TASK REGISTER 14
(4C)	76	SIGNED	4	TCRF	TASK REGISTER 15
(50)	80	SIGNED	4	TCR0	TASK REGISTER 0
(54)	84	SIGNED	4	TCR1	TASK REGISTER 1
(58)	88	SIGNED	4	TCR2	TASK REGISTER 2
(5C)	92	SIGNED	4	TCR3	TASK REGISTER 3
(60)	96	SIGNED	4	TCR4	TASK REGISTER 4
(64)	100	SIGNED	4	TCR5	TASK REGISTER 5
(68)	104	SIGNED	4	TCR6	TASK REGISTER 6
(6C)	108	SIGNED	4	TCR7	TASK REGISTER 7
(70)	112	SIGNED	4	TCR8	TASK REGISTER 8
(74)	116	SIGNED	4	TCR9	TASK REGISTER 9
VARIOUS CONTROL FIELDS					
(78)	120	SIGNED	4	TCRS (0)	RESTART INFORMATION
(78)	120	SIGNED	4		TASK TERMINATOR WORK AREA
(78)	120	SIGNED	4		IPW$$XTC ECB FOR DISPLAY SPOOL LST
(78)	120	BITSTRING	1	TCRX	RESTART FUNCTION INDEX
		...1		TCSP	"X'10'" ..SETUP REQUESTED
		...1 1...		TCCKP	"X'18'" ..CHECKPOINT REQUEST
		...1 11..		TCPAE	"X'1C'" ..POSITION AT END IF ERROR
(79)	121	ADDRESS	3		RESERVED FOR FUTURE USE
(79)	121	BITSTRING	1	TCCTRC	CURRENT TRC COMMAND CODE
(7A)	122	SIGNED	2	TCBL	BUFFER LENGTH
(78)	120	BITSTRING	1	TCRYFRB	FUNCT. REQ. BYTE OF CALLER
(79)	121	BITSTRING	1	TCRYTD	HELP FIELD USED BY RECOVERY
(7C)	124	BITSTRING	1	TCDT	DEVICE TYPE CODE
(7D)	125	BITSTRING	1	TCAT	ACCOUNT TRACE INDICATOR Used by the task terminator (TR) to determine the appropriate action in case of an I/O error on the account file. It can contain the following:
		.1..		TCJKB	"X'40'" .. PUT-ACCOUNT COMPLETE
		CHARACTER		TCJKL	"C'L' .. PUT-ACCOUNT ACTIVE @D35DI01
		CHARACTER		TCJKA	"C'A'" .. CALLER ACTIVE
		CHARACTER		TCJKC	"C'C'" .. CLOSE GET-MODE
		CHARACTER		TCJKE	"C'E'" .. ERASE ACCOUNT FILE
		CHARACTER		TCJKG	"C'G'" .. GET MODE
		CHARACTER		TCJKK	"C'K'" .. KEEP ACCOUNT FILE
		CHARACTER		TCJKO	"C'O'" .. OPEN GET-MODE
(7E)	126	BITSTRING	2	TCDE	PACKED DEVICE ADDRESS
(80)	128	SIGNED	4	TCRG	SAVE AREA FOR SERV. RTNS
(84)	132	SIGNED	4	TCRH	SAVE AREA FOR SERV RTNS
(88)	136	SIGNED	4	TC15	2ND BASE REG. SAVE AREA
WHENEVER A VSE/POWER SERVICE FUNCTION IS CALLED (EXCEPT TASK MANAGEMENT) REGISTER 9 IS SAVED IN TC09. REGISTER 9 IS THEN USED AS 2ND BASE REGISTER BY THE NUCLEUS ROUTINES. REGISTER 8 IS SAVED IN TC08 TO BE USED AS 3RD BASE.					
(8C)	140	SIGNED	4	TC08	REGISTER 8 SAVE AREA
(90)	144	SIGNED	4	TC09	REGISTER 9 SAVE AREA
LINKAGE REGISTER SAVE AREA					

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
THE FOLLOWING FIELDS RECORD THE CONTENTS OF THE GENERAL PURPOSE REGISTERS 14 THROUGH 9 WHENEVER ENTRY IS MADE BY THE TASK TO A VSE/POWER FUNCTION.					
(94)	148	CHAR-ACTER	56	TCSV (0)	REGISTER SAVE AREA
(94)	148	SIGNED	4		TASK CONTROL ADDRESS
(98)	152	SIGNED	4		PREVIOUS SAVE AREA ADDRESS
(9C)	156	SIGNED	4		SAVED REGISTER 14
(A0)	160	SIGNED	4		SAVED REGISTER 15
(A4)	164	SIGNED	4		SAVED REGISTER 0
(A8)	168	SIGNED	4		SAVED REGISTER 1
(AC)	172	SIGNED	4		SAVED REGISTER 2
(B0)	176	SIGNED	4		SAVED REGISTER 3
(B4)	180	SIGNED	4		SAVED REGISTER 4
(B8)	184	SIGNED	4		SAVED REGISTER 5
(BC)	188	SIGNED	4		SAVED REGISTER 6
(C0)	192	SIGNED	4		SAVED REGISTER 7
(C4)	196	SIGNED	4		SAVED REGISTER 8
(C8)	200	SIGNED	4		SAVED REGISTER 9
TRACE FACILITY SAVE AREA					
(CC)	204	SIGNED	4	TCTCWKP	TASK TRACE WORKAREA PNTR
(D0)	208	CHAR-ACTER		TCTCR (0)	TASK TRACE REG SAVEAREA
(D0)	208	SIGNED	4	TCTCRD	TASK TRACE REG 13
(D4)	212	SIGNED	4	TCTCRE	TASK TRACE REG 14
(D8)	216	SIGNED	4	TCTCRF	TASK TRACE REG 15
(DC)	220	SIGNED	4	TCTCR0	TASK TRACE REG 0
(E0)	224	SIGNED	4	TCTCR1	TASK TRACE REG 1
(E4)	228	SIGNED	4	TCTCR2	TASK TRACE REG 2
(E8)	232	SIGNED	4	TCTCR3	TASK TRACE REG 3
(EC)	236	SIGNED	4	TCTCR4	TASK TRACE REG 4
(F0)	240	SIGNED	4	TCTCR5	TASK TRACE REG 5
(F4)	244	SIGNED	4	TCTCR6	TASK TRACE REG 6
(F8)	248	SIGNED	4	TCTCR7	TASK TRACE REG 7
(FC)	252	SIGNED	4	TCTCR8	TASK TRACE REG 8
(100)	256	SIGNED	4	TCTCR9	TASK TRACE REG 9
IDUMP SAVE AREA					
WHENEVER THE IDUMP FUNCTION IS REQUESTED BY IPW$IDM, CALLER REGISTERS RE-R1 ARE SAVED IN TCIE-TCI1 NOTE - THIS AREA IS REUSED AS ECB LIST FOR IPW$WFM CALLS BY IPW$$QM(Q1), IPW$$T1, IPW$$XWE. IN $WFM STATE IDUMP IS NOT POSSIBLE EXCEPT DISPATCHER DETECTS DESTROYED TCB AND TERMINATES POWER					
(104)	260	SIGNED	4	TCIE	REGISTER E SAVE AREA
(108)	264	SIGNED	4	TCIF	REGISTER F SAVE AREA
(10C)	268	SIGNED	4	TCI0	REGISTER 0 SAVE AREA
(110)	272	SIGNED	4	TCI1	REGISTER 1 SAVE AREA
TASK MESSAGE INTERFACE					
(114)	276	ADDRESS	4	TCMW	ADDRESS OF MESSAGE TO BE ISSUED Consists of four bytes. The first byte contains the flag byte. The remaining three bytes contain the message address. The message address field contains the virtual address of the message control byte, that is, the byte that immediately precedes the text of the message to be output. X'80' .. HOLD MESSAGE CONTROL BLOCK X'40' .. REGISTER 5 CONTAINS TCB ADDRESS X'20' .. MESSAGE IS IN NMR FORMAT
		...1		TCDNCM	"X'10'" .. DO NOT COMPRESS MESSAGE
		.... 1...		TCPCOP	"X'08'" .. PASS TO CENTRAL OPERATOR
		.... .1..		TCDNMM	"X'04'" .. DO NOT MODIFY MESSAGE
(118)	280	ADDRESS	4	TCAW	ADDRESS OF CALLERS REPLY AREA

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
WTO/WTOR/DOM INTERFACE: WTO/WTOR OUTPUT					
(11C)	284	BITSTRING	4	TCMID	MESSAGE ID
(120)	288	ADDRESS	4	TCMRECB	WTOR REPLY ECB
WTO/WTOR/DOM INTERFACE: WTO/WTOR/DOM INPUT - SET BY POWER:					
(124)	292	BITSTRING	4	TCMRT	MESSAGE ROUTING CODE
(128)	296	BITSTRING	2	TCMDC	MESSAGE DESCRIPTOR CODE
(12A)	298	BITSTRING	1	TCF18	FLAG BYTE 18
		1...		TCF1863	"X'80'" .. TAPE FROM VERS. 5.2-6.3
		.1..		TCF18WW	"X'40'" .. $$XW WAIT ON SEG. POSTING
					.. $$XW POST WAITING $$XW
		..1.		TCF18CE	"X'20'" .. CONNECTED MSG ERROR
		...1		TCF18I1	"X'10'" .. DO IDUMP ONCE PER TCB
(12B)	299	BITSTRING	1	TCF19	FLAG BYTE 19
(12C)	300	BITSTRING	4	TCMNRT	NEG ROUTING CODE(DON"T WANT)@D61CDSW
(130)	304	BITSTRING	4	TCMRTDF	DEFAULT MSG ROUTING CODE
(134)	308	ADDRESS	4	TCMDOM	DOM MESSAGE ID
(138)	312	ADDRESS	4	TCMCID	COMMAND CONNECT MESSAGE ID
AR COMMAND OUTPUT(INPUT TO WTO/WTOR IF CMD RESP)					
(13C)	316	BITSTRING	8	TCMCART	AR MESSAGE TOKEN (CART)
(144)	324	BITSTRING	4	TCMCOID	AR CONSOLE ID
MISCELLANEOUS					
(148)	328	BITSTRING	2	TCMPID	PARTITION ID
(14A)	330	BITSTRING	1	TCMFLG	MESSAGE FLAGS
		1...		TCMFAR	"X'80'" .. VSE/AF CMD
		.1..		TCMFUR	"X'40'" .. VSE/AF CMD USER CONSOLE
		..1.		TCMFCUP	"X'20'" .. CLOSE UP CONN'D MSGS
		...1		TCMFCFM	"X'10'" .. ISSUE 1ST CONN'D MESSAGE
		.... 1...		TCMFICM	"X'08'" .. ISSUE CONNECTED MESSAGE
		.... .1..		TCMFCEX	"X'04'" .. CONNECTED MSG EXISTS
(14B)	331	BITSTRING	1		UNUSED
(14C)	332	ADDRESS	4	TCVD	SAVED PTR(LINK-REG-SV-AREA)
(150)	336	ADDRESS	4	TC1Q40	MSG 1Q40A MSG ID FOR DOM
(150)	336	ADDRESS	4	TC1Q38	MSG 1Q38A MSG ID FOR DOM
(150)	336	ADDRESS	4	TC1QD6	MSG 1QD6I MSG ID FOR DOM
(154)	340	ADDRESS	4		UNUSED
(158)	344	ADDRESS	4	TCPFTWA	TAPE-WORKAREA NOT TO REL.
(15C)	348	ADDRESS	4	TCPFPWA	PRT/PUN-WA NOT TO RELEASE
(160)	352	SIGNED	2	TCPFCU	CUU OF PSTOP FORCE CMD
(162)	354	SIGNED	2	TCPFRC	RC OF PSTOP FORCE CMD
DATA FILE CONTROL WORDS INPUT/OUTPUT REQUEST WORD					
(164)	356	ADDRESS	4	TCDW	RELATIVE DBLK NUMBER
(168)	360	ADDRESS	4	TCDV	VIRTUAL DATA AREA ADDRESS
(16C)	364	ADDRESS	2	TCDL	DATA AREA LENGTH
(16E)	366	BITSTRING	1		OPERATION CODE
(16F)	367	BITSTRING	1		RESERVED
BLOCKING CONTROL WORDS					
(170)	368	SIGNED	4	TCBC	RESIDUAL BLOCK COUNT
(174)	372	ADDRESS	4	TCPR	PREVIOUS/NEXT RECORD ADDRESS
(178)	376	ADDRESS	4	TCASPB	ADDR OF SPOOL BLOCK
RECORD CONTROL WORD (RCW)					
(17C)	380	CHAR- ACTER	8	TCRW (0)	RECORD CONTROL WORD
(17C)	380	BITSTRING	1	TCCC	RECORD COMMAND CODE X'FF' .. CONTROL RECORD X'FE' .. NEW FORMS
(17D)	381	ADDRESS	3	TCRV	RECORD ADDRESS (VIRTUAL)
(180)	384	BITSTRING	1	TCGP	GENERAL PURPOSE BYTE
		....		GPNR	"X'00'" .. NORMAL RECORD
		.1..		GPER	"X'40'" .. EXTENDED RECORD

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
		..1.		GPDE	"X'20'" .. END OF 3540 DATA
		...1		GPEB	"X'10'" .. END OF BLOCK
					X'08' .. >>NOT USABLE<< (OLD GPBR FIELD. X
		.... .1..		GPED	"X'04'" .. END OF DATA (=EOF)
		.... ..1.		GPRD	"X'02'" .. 3540 DATA RECORD
		1		GPDR	"X'01'" .. LINE PRINT/CARD MOVE/DATA
(181)	385	BITSTRING	1	TCG2	GENERAL PURPOSE BYTE TWO
		1...		TCGJ	"X'80'" .. JOB HEADER RECORD
		.1..		TCGT	"X'40'" .. JOB TRAILER RECORD
		..1.		TCGD	"X'20'" .. DATASET HEADER RECORD
		...1		TCSMR	"X'10'" .. STREAM MODE RECORD (CPDS)
		.... 1...		TCG08	"X'08'" .. UNUSED
		.... .1..		TCFFM	"X'04'" .. FIXED FORMAT MESSAGE REC
		.... ..1.		TCASA	"X'02'" .. ASA DATA RECORD
(182)	386	SIGNED	2	TCRL	RECORD LENGTH
(184)	388	BITSTRING	1	TCG3	GENERAL PURPOSE BYTE THREE
		1...		TCSEP	"X'80'" .. SEPARATOR PAGE/CARD REC
		.1..		TCESEP	"X'40'" .. END SEP. PAGE/CARD REC
		..1.		TCEOC	"X'20'" .. END OF COPY RECORD
		...1		TCNRI	"X'10'" .. NO RECORD INCREMENT
(185)	389	BITSTRING	1	TCG4	GENERAL PURPOSE BYTE FOUR
		1...		TCILC	"X'80'" .. LINE COUNT TO BE INCRE"D @D23CDWS
		.1..		TCIPC	"X'40'" .. PAGE COUNT TO BE INCRE"D @D23CDWS
		..1.		TCSGN	"X'20'" .. SUPRESS GET NEXT CCW
(186)	390	BITSTRING	1	TCDVEB	'DEVICE END' OCCURRED ('FF')
(187)	391	BITSTRING	1		RESERVED FOR FUTURE USE
(188)	392	SIGNED	4	TCLRNO	LOGICAL RECORD NUMBER
QUEUE FILE CONTROL WORDS					
(18C)	396	ADDRESS	4	TCQW	RELATIVE QUEUE REC NUMBER
(190)	400	ADDRESS	4	TCQV	VIRTUAL SPACE ADDRESS
(194)	404	ADDRESS	2		QUEUE REC LENGTH - NOT USED
(196)	406	BITSTRING	1		OPERATION CODE
(197)	407	BITSTRING	1		RESERVED
TAPE SPOOLING CONTROL INFORMATION					
(198)	408	BITSTRING	8	TCTS (0)	TAPE REQUEST WORD
(198)	408	BITSTRING	1	TCTF	FUNCTION BYTE USED FOR TAPE
(199)	409	ADDRESS	3	TCTA	ADDRESS OF TAPE CTRL BLOCK
(19C)	412	BITSTRING	4	TCTDES (0)	TAPE UNIT DESCRIPTORS
(19C)	412	BITSTRING	1	TCTM	INDICATE TAPE MODE (DENS.)
(19D)	413	BITSTRING	1	TCTDT	TAPE DEVICE TYPE
(19E)	414	CHAR- ACTER	2	TCTU	TAPE PROG.LOGICAL UNIT(PUU)
(1A0)	416	SIGNED	4	TCPU	PHYS.UNIT OR PUB ENTRY ADDR.
VARIOUS CONTROL FIELDS					
(1A4)	420	BITSTRING	1	TCF2	FLAG BYTE 2
		1...		TCT2S	"X'80'" .. 2.ND TIME SWITCH IPW$$TR
		.1..		TCERT	"X'40'" .. EXECUTION READER TASK ID
		..1.		TCWOP	"X'20'" .. WRITER-ONLY PARTITION ID
		...1		TCOFF	"X'10'" .. POFFLOAD TASK
		.... 1...		TCSLI	"X'08'" .. SLI IN PROCESS
		.... .1..		TCJBP	"X'04'" .. JOB STMT PROCESSING
		.... ..1.		TCLTP	"X'02'" .. LST STMT PROCESSING
		1		TCPUP	"X'01'" .. PUN STMT PROCESSING
(1A5)	421	BITSTRING	1	TCF3	FLAG BYTE 3
		1...		TCHCPY	"X'80'" .. HORIZONTAL COPY IND
		.1..		TCSGM	"X'40'" .. SEGMENTATION REQUIRED
		..1.		TCDFT	"X'20'" .. DEFAULT REQUIRED FOR 3800
		...1		TCIDH	"X'10'" .. INSERT DATASET HDR REC
		.... 1...		TCUNN	"X'08'" .. Q REC WITH UNKNOWN NODEID
		.... .1..		TCOTV	"X'04'" .. OLD TAPE VERSION 1
		.... ..1.		TCNRW	"X'02'" .. NO REWIND REQUESTED
		1		TCSIN	"X'01'" .. SYSIN MODE REQUESTED

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(1A6)	422	BITSTRING	1	TCF4	FLAG BYTE 4
		1...		TCCSF	"X'80'" .. DOS STMT CONTINUED (1ST)
		.1..		TCCSC	"X'40'" .. DOS STMT CONTINUED
		..1.		TCCSJ	"X'20'" .. JECL STMT CONTINUED ERROR
		...1		TCPGEP	"X'10'" .. PAGE POSITIONING WANTED
		.... 1...		TCDSHR	"X'08'" .. DSHR RECORD BUILT
		.... .1..		TCVMW	"X'04'" .. VM WRITER TASK
		.... ..1.		TCDST	"X'02'" .. DEVICE SERVICE TASK
		1		TCXPT	"X'01'" .. CROSS PARTITION USER TASK
(1A7)	423	BITSTRING	1	TCF5	FLAG BYTE 5
		1...		TCDNA	"X'80'" .. DO NOT WRITE ACCOUNT REC
		.1..		TCDNS	"X'40'" .. DO NOT SPOOL RECORD
		..1.		TCIQM	"X'20'" .. IGNORE QUEUE MANAGEMENT
		...1		TCIQJ	"X'10'" .. IGNORE FOLLOWING JOBS
		.... 1...		TCFRNW	"X'08'" .. NO-WAIT WANTED
		.... .1..		TCLPOS	"X'04'" .. LINE POSITIONING WANTED
		.... ..1.		TCCAR	"X'02'" .. ASA CONVERSION REQUESTED
		1		TCSPW	"X'01'" .. SPOOLED WRITER TASK IND.
(1A8)	424	BITSTRING	1	TCF6	FLAG BYTE 6
		1...		TCFEW	"X'80'" .. 'FE' RECORD WRITTEN
		.1..		TCFRY	"X'40'" .. QUEUE FILE RECOVERY ACTIV
		..1.		TCFIOH	"X'20'" .. I/O ERROR HANDLER ACTIVE
		...1		TCFSEP	"X'10'" .. SEP PAGES/CARDS WANTED
		.... 1...		TCFNOS	"X'08'" .. NO SEP PAGES/CARDS WANTED
		.... .1..		TCFIOD	"X'04'" .. DATA FILE I/O ERROR PROC
		.... ..1.		TCFNSC	"X'02'" .. Q-ENTRY: NO STATUS CHANGE
		1		TCFDCD	"X'01'" .. DO NOT ALTER DISPOSITION
(1A9)	425	BITSTRING	1	TCF7	FLAG BYTE 7
		1...		TCFUQE	"X'80'" .. UNCHAIN QUEUE ENTRY
		.1..		TCCLNU	"X'40'" .. CNTL BLOCK LOCKED BY $$NU
		..1.		TCFSKP	"X'20'" .. SKIP TO CH1 INSERTION
		...1		TCRLCK	"X'10'" .. REVERSE LOCKING ORDER
		.... 1...		TCVS2	"X'08'" .. TAPE, POWER VERSION 2
		.... .1..		TCXLR	"X'04'" .. JOBEXIT WORK AREA RESER.
		.... ..1.		TCFMSG	"X'02'" .. SEND SIGNAL ATTENTION
		1		TCFNOF	"X'01'" .. DON'T FLUSH JOB
(1AA)	426	BITSTRING	1	TCF8	FLAG BYTE 8
		1...		TCF8NM	"X'80'" .. NO MESSAGE MODIFICATION
		.1..		TCF8WXW	"X'40'" .. XR IS WAITING FOR XW
		..1.		TCF8NW	"X'20'" .. NO REAL STORAGE WAIT OPT. .. NO WORK-SPACE OBTAINED
		...1		TCF8DY	"X'10'" .. DYNAMIC EXEC. PROCESSOR
		.... 1...		TCF8DS	"X'08'" .. DYNAMIC READER STARTING
		.... .1..		TCF8LO	"X'04'" .. LOAD MACRO RC INDICATOR
		.... ..1.		TCF8MS	"X'02'" .. MODE SET/REQUEST DONE
		1		TCF8HP	"X'01'" .. HIGH PERFORMANCE OPTION
(1AB)	427	BITSTRING	1	TCF9	FLAG BYTE 9
		1...		TCF9TA	"X'80'" .. TASK MAY USE ACC REG
		.1..		TCF9UA	"X'40'" .. ACCESS REG ARE USED
		..1.		TCF9AM	"X'20'" .. ACCESS-REG MODE SET ON
		...1		TCF9CO	"X'10'" .. $$PS CALL BY CENT. OP.
		.... 1...		TCF9WA	"X'08'" .. DST: WAIT IF SOA
		.... .1..		TCF9SD	"X'04'" .. INVALID SER DBLK DETECTED
		.... ..1.		TCF9NM	"X'02'" .. NO MSG MODIFICATION
		1		TCF9RQ	"X'01'" .. QUEUE SET RESERVED
(1AC)	428	BITSTRING	1	TCF10	FLAG BYTE 10
		1...		TCF10V4	"X'80'" .. TAPE FROM 4.X. OR 5.1
		.1..		TCF10UI	"X'40'" .. USER INFO IN MSG
		..1.		TCF10RX	"X'20'" .. RUNNING TASK DETECTED BY TPEND EXIT, TESTED BY $$SN.
		...1		TCF10QC	"X'10'" .. QCM SUPPPORT EXISTS
		.... 1...		TCF10FN	"X'08'" .. F.F. NMR BUILT
		.... .1..		TCF10DL	"X'04'" .. DROP LAST SEPARATOR PAGE
		.... ..1.		TCF10WR	"X'02'" .. WRITE OCCURED IN XW/XWE
		1		TCF10IS	"X'01'" .. USE IDENTICAL SEP. PAGES

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(1AD)	429	BITSTRING	1	TCF11	FLAG BYTE 11
		1...		TCF11NL	"X'80'" .. NO VSCB LOCKING ($UNV)
		.1..		TCF11XF	"X'40'" .. USER EXIT FAILED
		..1.		TCF11JC	"X'20'" .. CONT JCL MODE - SKIP JECL
		...1		TCF11NT	"X'10'" .. TRACING NOT ALLOWED
		.... 1...		TCF11TR	"X'08'" .. TRACING WORKAREA REAL
		.... .1..		TCF11SP	"X'04'" .. STOP POST-SLOT PROCESS
		.... ..1.		TCF11SI	"X'02'" .. SHOW IGNORED RECORDS
		1		TCF11SF	"X'01'" .. SEGMENTATION FAILED
(1AE)	430	BITSTRING	1	TCF12	FLAG BYTE 12
		1...		TCF12SD	"X'80'" .. REG 13 SAVED IN $MS
		.1..		TCF12DI	"X'40'" .. ISSUE DUMMY I/O TO JCL
		..1.		TCF12FF	"X'20'" .. FIXED FORMAT MSG QUEUED
		...1		TCF12DF	"X'10'" .. DEFAULT FCB USED ($XWE)
		.... 1...		TCF12LP	"X'08'" .. LTAB IN * $$ LST ($XWE)
		.... .1..		TCF12NE	"X'04'" .. DO NOT PASS TO RDR-EXIT
		.... ..1.		TCF12R1R	"X'02'" .. READ 1 RECORD FOR PHYSRDR
		1		TCF12JOB	"X'01'" .. // JOB ACTIVE IN LR JOBXT
(1AF)	431	BITSTRING	1	TCF13	FLAG BYTE 13 (RES.FOR GCM)
		1...		TCF13QQ	"X'80'" .. QUEUE FIX FORM MSG REQ.
		.1..		TCF13GP	"X'40'" .. TASK IN GCM-WAIT-PEND ST.
		..1.		TCF13CQ	"X'20'" .. TASK IN COM.QU. STATE
		...1		TCF13DQ	"X'10'" .. TASK IN DOUBL. QU. STATE
		.... .1..		TCF13GE	"X'04'" .. GCM-WAIT PEND TERM STATE
		.... ..1.		TCF13ME	"X'02'" .. GCM-WAIT MSG EVENT
		1		TCF13KP	"X'01'" .. KEEP GCM WAIT IF PEND
(1B0)	432	BITSTRING	1	TCF14	FLAG BYTE 14
		1...		TCF14KN	"X'80'" .. IPWSEGM KEEP=NO PROCESS
		.1..		TCF14KY	"X'40'" .. IPWSEGM KEEP=YES PROCESS
		..1.		TCF14PL	"X'20'" .. OUTPUT IGNORED (LST0DAT)
		...1		TCF14PI	"X'10'" .. OUTPUT IGNORRED
		.... 1...		TCF14PP	"X'08'" .. OUTPUT PURGED
		.... .1..		TCF14Q5	"X'04'" .. $$XW TO ISSUE DEBUG 1Q53I
		.... ..1.		TCF1452	"X'02'" .. $$XWE 1Q52I ISSUED ONCE
(1B1)	433	BITSTRING	1	TCF15	FLAG BYTE 15
		1...		TCF15US	"X'80'" .. USER EXIT SET STOPCODE S
		.1..		TCF15IG	"X'40'" .. IGNORE 1ST SEGMENT I/O
		..1.		TCF15DU	"X'20'" .. DYNAM PARTITION WAS UP
		...1		TCF15SU	"X'10'" .. IPW$GAM SUB=YES CALLED
		.... 1...		TCF15MR	"X'08'" .. C-MSG FOR PRELEASE
		.... .1..		TCF15PF	"X'04'" .. PSTOP FORCE ISSUED
		.... ..1.		TCF15PN	"X'02'" .. PSTOP FORCE I/F $CP/$NU
(1B2)	434	BITSTRING	1	TCF16	FLAG BYTE 16
		1...		TCF16TI	"X'80'" .. TAPE I/O BUSY
		.1..		TCF16PI	"X'40'" .. PRT/PUN I/O BUSY
		..1.		TCF16NP	"X'20'" .. NON-PARALLEL MODE ACTIVE
		...1		TCF16P3	"X'10'" .. WLST/WPUN TASK FOR P390 4 FLAGS FOR PAGE COUNT STATE:
		.... 1...		TCF16LM	"X'08'" .. LINE MODE (DEFAULT)
		.... .1..		TCF16LMI	"X'04'" .. LINE MODE IDM/IMM RECV'D
		.... ..1.		TCF16PM	"X'02'" .. PAGE MODE
		1		TCF16PM8	"X'01'" .. PAGE MODE '8B' RECEIVED
(1B3)	435	BITSTRING	1	TCF17	FLAG BYTE 17
		1...		TCF17TBL	"X'80'" .. BAM LABELED TAPE
		.1..		TCF17TBN	"X'40'" .. BAM UNLABELED TAPE
		..1.		TCF17TBW	"X'20'" .. BAM WRITE MODE
		...1		TCF17TMS	"X'10'" .. TMS TAPE HANDLG(RESERVED)
		.... 1...		TCF17PA	"X'08'" .. PICKUP HAS ACTIVE ENTRY
		.... .1..		TCF17LTA	"X'04'" .. BAM HOLDS LTA DURING REQ.
		.... ..1.		TCF17WFW	"X'02'" .. $$XR WAITING ON BAM OPEN
		1		TCF17A31	"X'01'" .. PAGE FAULT IN AMODE 31
(1B4)	436	BITSTRING	1	TCF20	FLAG BYTE 20
		1...		TCF20BC	"X'80'" .. SAS BROWSE IN CREATION
		.1..		TCF20DT	"X'40'" .. OPEN IJDTEST IN PROGRESS
		..1.		TCF20FMT	"X'20'" .. FORMATTING ADD. EXTEMT

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
		...1		TCF201ST	"X'10'" .. OPEN IJDTEST ERR.1ST TIME
(1B5)	437	BITSTRING	1	TCF21	FLAG BYTE 21
		1...		TCF21XXX	"X'80'" .. UNUSED
(1B6)	438	BITSTRING	6		.. UNUSED
(1BC)	444	SIGNED	4	TCVEB (0)	ECB FOR VTAM REQUESTS
(1BC)	444	SIGNED	4	TCFEB (0)	ECB FOR FORMAT DATA FILE
(1BC)	444	BITSTRING	2		RESERVED - DO NOT USE !
(1BE)	446	BITSTRING	1	TCVEP	EVENT POST BYTE FOR VTAM
(1BE)	446	BITSTRING	1	TCFEP	EVENT POST BYTE FOR FORMAT DFILE
		1...		TCVEO	"X'80'" ..EVENT POST BIT ON
		1...		TCFEO	"X'80'" ..EVENT POST BIT ON
(1BF)	447	BITSTRING	1		RESERVED - DO NOT USE !
(1C0)	448	ADDRESS	4	TCGDS	ADDR OF GENERATED DSHR
(1C4)	452	ADDRESS	4	TCCMRG	COMREG ADDR IF EXEC TASK
(1C8)	456	SIGNED	4	TC3E	ADDR OF TCB EXTENSION AREA OR ADDR OF WORK SPACE OR DSHR
(1CC)	460	SIGNED	4	TCHD	HEAD PTR VIRT STORAGE CHAIN
(1D0)	464	SIGNED	4		TAIL PTR VIRT STORAGE CHAIN
(1D4)	468	ADDRESS	4	TC3W	POINTER 3540 WORKSPACE
(1D8)	472	SIGNED	4	TCXWA	ADDRESS TO EXIT WORK AREA
(1DC)	476	SIGNED	4	TCJHR	PTR TO JHR (USED BY $LR)
(1E0)	480	SIGNED	2	TCXWAL	LENGTH OF EXIT WORK AREA
(1E2)	482	SIGNED	2	TCQCQW	QUEUE REC. OF QC.. USED BY $SQ IN CASE OF SOD FOR QCA
(1E4)	484	SIGNED	4	TCLRWA	LOGICAL READER WORK AREA
(1E8)	488	SIGNED	4	TCRVAL	RESTART PAGE/LINE/RECORD CNT
(1EC)	492	SIGNED	4	TC0EEX	RETURN ADDRESS OF USER EXIT
(1F0)	496	SIGNED	4	TCNR2W	PTR TO NR2 WORKAREA
	LOGICAL I/F AND FUNCTION REQUEST BYTES				
(1F4)	500	BITSTRING	1	TCLIFB	LOG. INTERFACE FUNCTION BYTE
		1		TCLIOP	"X'01'" .. PRE-OPEN OUTPUT QUEUE
		.... ..1.		TCLIOF	"X'02'" .. FINAL OPEN OUTPUT QUEUE
		11		TCLILO	"X'03'" .. LOCATE QUEUE ENTRY
		.... .1..		TCLIOR	"X'04'" .. OPEN-RESTART QUEUE ENTRY
		.... .1.1		TCLIRS	"X'05'" .. RESTART QUEUE ENTRY
		.... .11.		TCLISG	"X'06'" .. SEGMENT OUTPUT Q' ENTRY
		.... .111		TCLIFL	"X'07'" .. FLUSH OUTPUT QUEUE ENTRY
		.... 1...		TCLICH	"X'08'" .. CHECKPOINT QUEUE ENTRY
		.... 1..1		TCLICL	"X'09'" .. CLOSE OUTPUT QUEUE ENTRY
		.... 1.1.		TCLISP	"X'0A'" .. SPOOL RECORD
		.... 1.11		TCLIAQ	"X'0B'" .. ADD TO CLASS CHAIN
		.... 11..		TCLICO	"X'0C'" .. CLOSE WITHOUT JTR
		.... 11.1		TCLIRL	"X'0D'" .. READ LOCATED DATA
		.... 111.		TCLIOFJ	"X'0E'" .. FINAL OPEN OUTPUT QUEUE WITH JOBNUMBER SUPPLIED
		 1111		TCLIFLW	"X'0F'" .. FLUSH OUTPUT QUEUE ENTRY WITHOUT MESSAGE
		...1		TCLIOFB	"X'10'" .. FINAL OPEN OUTPUT QUEUE WITH JOBNUMBER SUPPLIED, NO BLANK TRUNCATION
(1F5)	501	BITSTRING	1	TCLIRC	LOG. INTERFACE RETURN CODE
		....		TCLROK	"X'00'" .. OK
		1		TCLRWS	"X'01'" .. WARNING-SPOOL CC,IGNORED
		.... ..1.		TCLRLO	"X'02'" .. ERROR - LOCATE FAILED
		11		TCLRRR	"X'03'" .. RESTART ERROR, WRONG REC
		.... .1.1		TCLRCF	"X'05'" .. CHECKPOINTING FAILED
		.... .11.		TCLRSE	"X'06'" .. SPOOLING ERROR (QUALIFIER IN 2ND BYTE)
		.... .111		TCLRSDA	"X'07'" .. SPOOLING ERROR (SOD+SOA)
		.... 1...		TCLROR	"X'08'" .. OPEN RESTART ERROR
		.... 1..1		TCLRWC	"X'09'" .. CHECKPOINT OD ALTERED
(1F6)	502	BITSTRING	1	TCLISR	2ND LOG. INTERFACE RC
(1F7)	503	BITSTRING	1	TCFRB1	FUNCTION REQUEST BYTE 1
		1			"X'01'" .. UNUSED
------------ SET BY $$DS -----------------------------------					

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
		.... ..1.		TCFRBR	"X'02'" .. RESTART QUEUE SET SPOOL'G
		11		TCFRBU	"X'03'" .. UPDATE SPOOL ENV BLOCK
		.... .1..			"X'04'" .. UNUSED
		.... .1.1		TCFRBP	"X'05'" .. RESET SPOOLING POINTERS
------------ SET BY $$CS ------------------------					
		.... .11.		TCFRQS	"X'06'" .. SCAN CLASS CHAIN (IGNORE LIVE BIT SETTING)
------------ SET BY $$I4/$$TI -----------------------------					
		.... .111		TCFFUL	"X'07'" .. PERFORM FULL RECOVERY
------------ SET BY $IQS/$AQS/$DQS/$FQS --------------------					
		.... 1...		TCFNOL	"X'08'" .. DO NOT UNLOCK DMB
------------ SET BY $ICP ----------------------------------					
		.... 1..1		TCFRIC	"X'09'" .. DO NO AUTHORITY CHECK
(1F8)	504	BITSTRING	1	TCFRCT	FUNCTION RETURN CODE
------------ SET BY $$PA/$$PF, $$PD AND $$RQ --------------					
		1		TCFSOD	"X'01'" .. SHORT-ON-DASD SPACE (SOD)
		.... ..1.		TCFSOA	"X'02'" .. SHORT-ON-ACCOUNT FILE
------------ SET BY $$NQ (GET NEXT QUEUE SET) -------------					
		11		TCFRNF	"X'03'" .. QUEUE SET NOT FOUND
		.... .1..		TCFRBY	"X'04'" .. QUEUE SET MARKED ACTIVE
		.... .1.1		TCFRQP	"X'05'" .. QUEUE SET PROTECTED
		.... .11.		TCFRNE	"X'06'" .. QUEUE SET NOT ELIGIBLE
		.... .111		TCFRSM	"X'07'" .. FOUND, BUT JOB SUFF MISS
		.... 1.11		TCFRQN	"X'0B'" .. INVALID Q-RECORD NUMBER
		.... 11..		TCFRSA	"X'0C'" .. SECURITY ACCESS VIOLATION
------------ SET BY $$DS (DATA MANAGEMENT SERVICES) --------					
		.... 1...		TCFRIR	"X'08'" .. INCORRECT RECORD (CC)
		.... 1..1		TCFRRS	"X'09'" .. INC. RESTART RECORD NO.
------------ SET BY $$NQ (GET NEXT QUEUE SET) -------------					
		.... 1.1.		TCFRUM	"X'0A'" .. QUEUE SET USER MISMATCH
(1F9)	505	BITSTRING	1	TCFRB2	FUNCTION REQUEST BYTE 2
------------ SET BY $ICP ----------------------------------					
		1		TCFPCE	"X'01'" .. PASS PCE
		.... ..1.		TCFVER	"X'02'" .. PASS VERIFIED CLASS TABLE
		11		TCFNLK	"X'03'" .. DO NOT LOCK DPCB
		.... 11.1		TCFOCP	"X'0D'" .. PASS CMD FROM PERM. CP
------------ SET BY $ITQ ----------------------------------					
		.... .1..		TCFAWF	"X'04'" .. ADD TO WFR-SUBQUEUE
		.... .1.1		TCFDWF	"X'05'" .. DEL FROM WFR-SUBQUEUE
		.... .11.		TCFIWF	"X'06'" .. INIT THE WFR-SUBQUEUE
------------ SET BY $$XTG FOR $$LW ------------------------					
		.... .111		TCFCKI	"X'07'" .. EXTENDED CKP INFO PASSED
------------ SET BY $ICP ----------------------------------					
		.... 1...		TCFDCK	"X'08'" .. DELETE CKP INFO
		.... 1..1		TCFDQN	"X'09'" .. PASS DIRECT Q-REC NUMBER
------------ SET BY $IIS ----------------------------------					
		.... 1.1.		TCFDEL	"X'0A'" .. 'DIRECT' ELIGIB. CHECK
------------ SET BY $$LO ----------------------------------					
		.... 1.11		TCFRBL	"X'0B'" .. LOCATE QUEUE SET
		.... 11..		TCFSCT	"X'0C'" .. INFORM $$DQ TO SET QROTC
	CONTINUATION OF VARIOUS CONTROL FIELDS				

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(1FA)	506	BITSTRING	1	TCRSCO	REMAINING COPY NUMBER

THE FOLLOWING FIELD IS USED BY THE EXECUTION PROCESSOR ROUTINES (IPW$$XR & IPW$$XW) TO INDICATE THE DETAIL REASON CODE FOR MESSAGE 1R30I.

(1FA)	506	BITSTRING	1	TCERC	DETAIL REASON CODE
		1		TCERCE	"X'01'" .. CCW WITH DC, IDAL
		.... ..1.		TCERCI	"X'02'" .. INVALID CCW OP CODE
		11		TCERCO	"X'03'" .. CCW OUTSIDE OF PARTITION
		.... .1..		TCERCD	"X'04'" .. DATA AREA OUTSIDE OF PART
		.... .1.1		TCERCZ	"X'05'" .. WRONG RECORD LENGTH
		.... .11.		TCERCW	"X'06'" .. CCW ¬ ON D-WORD BOUNDARY
		.... .111		TCERC3	"X'07'" .. 3540 READ OUT OF SEQUENCE
		.... 1...		TCERCU	"X'08'" .. UNKNOWN CHANNEL SPECIFIED
		.... 1..1		TCERCF	"X'09'" .. WRONG FCB IMAGE
		...1		TCERCJ	"X'10'" .. WRONG JECL STATEMENT
		...1 ...1		TCERCT	"X'11'" .. MORE THAN 255 TIC'S
		...1 ..1.		TCERF1	"X'12'" .. FORMAT 1 CCW USED
		...1 ..11		TCERCR	"X'13'" .. CCB INDICATES EXCP REAL

THE FOLLOWING DEFINITIONS ARE USED BY THE EXECUTION PROCESSOR ROUTINE (IPW$$XW) TO INDICATE THE DETAIL REASON CODE FOR MESSAGE 1Q54I.

		1		TCERCN	"X'01'" .. PHASE NOT FOUND
		.... ..1.		TCERCL	"X'02'" .. INCORRECT PHASE LENGTH
		11		TCERCP	"X'03'" .. INVALID FCB PHASE NAME PREFIX (3800)
		.... .1..		TCERCC	"X'04'" .. INVALID CHANNEL
		.... .1.1		TCERCV	"X'05'" .. INVALID FCB END COND.
		.... .11.		TCERCG	"X'06'" .. WRONG LINE/PAGE FLAG 3800
		.... .111		TCERNF	"X'07'" .. LOADING NEW FORMAT FCB ON A NON D/T4248.
(1FB)	507	BITSTRING	1	TCLRRL	JOB REC LEN, 1ST SYSRDR DEV
(1FC)	508	SIGNED	4	TCPL	ADDR(SPOOL MGMT PARM LST) X'FF' .. SPOOL MGR SPL PRESENT
(200)	512	CHAR-ACTER	8	TCSECAU	SECURITY OWNING USERID
(208)	520	BITSTRING	1	TCSECFG	SECURITY FLAGS
(209)	521	BITSTRING	3		UNUSED
(20C)	524	ADDRESS	4	(5)	RESERVED

GENERAL TASK WORK AREA

THE FOLLOWING 88 BYTES ARE USED AS A GENERAL-PURPOSE WORK AREA, WHICH MAY BE BROKEN INTO FIELDS AS IS REQUIRED BY EACH SPECIFIC TASK.

(220)	544	SIGNED	4	(0)	
(220)	544	BITSTRING	32	TCGW	WORK AREA - USED BY LOGICAL ROUTINES, MAY NOT BE REUSED BY ANY TASK USING LOG. RTNS
(240)	576	SIGNED	4	TCW1	WORK WORD 1
(244)	580	SIGNED	4	TCW2	WORK WORD 2
(248)	584	SIGNED	4	TCW3	WORK FULLWORD 3
(24C)	588	SIGNED	4	TCW4	WORK FULLWORD 4
(250)	592	SIGNED	4	(4)	RESERVED
(260)	608	BITSTRING	16	TCGW2	WORK AREA 2
(270)	624	BITSTRING	72	TCGW3	WORK AREA 3

LOGICAL WRITER RE-DEFINITION

(220)	544	SIGNED	4	LWFI	FORMS IDENTIFICATION
(224)	548	BITSTRING	1	LWNC	COPY/TRANSMISSION COUNTER
(225)	549	ADDRESS	1	LWCSNO	CURR NUMBER FOR SEP PAGES
(226)	550	BITSTRING	1	LWFLG	LOG. WRITER FLAG BYTE
		1...		LWFEJ	"X'80'" ..RESTART EOJ INDICATOR
		.1..		LWRBA	"X'40'" ..RESTART BACKWARDS ACTIVE
		..1.		LWLRR	"X'20'" ..LOCATE RESTART RECORD ACT
		...1		LWTEO	"X'10'" ..UPD EXTRA AND TOTAL ONLY
		.... 1...		LWJHR	"X'08'" .. JHR ALREADY PASSED

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
		.... .1..		LWHCPY	"X'04'" .. HORIZONTAL COPY ON SET
(227)	551	BITSTRING	1	LWFT	START SEPARATION SWITCH
(228)	552	SIGNED	4	LWAW	ACCOUNT COUNTER WS PTR
(22C)	556	BITSTRING	1	LWLC (0)	LAST COMMAND CODE
(22C)	556	ADDRESS	4	LWAD	ADDRESS OF BUFFER FOR SEP
(230)	560	BITSTRING	1	LWIO	USED FOR SEP PAGES PROC
(231)	561	BITSTRING	1	LWSW	LOGICAL WRITER SWITCHES
		1...		LWFISW	"X'80'" .. DEFAULT FORMS ID SET
		.1..		LWFLSW	"X'40'" .. DEFAULT FLASH ID SET
		..1.		LWPSSW	"X'20'" .. DEFAULT BURST SET
(232)	562	BITSTRING	1	PPEB	EMPTY BLOCK INDICATOR $$PP
(233)	563	BITSTRING	1	LWPS	PAPER STATUS (3800 ONLY)
(234)	564	CHARACTER	4	LWFH	FLASH IDENTIFICATION
(238)	568	SIGNED	4	LWQRR	POSSIBLE RESTART INFO
(23C)	572	BITSTRING	1	LWQRX	RESTART FUNCTION INDEX
(23D)	573	BITSTRING	3		RESERVED FOR FUTURE USE
(270)	624	SIGNED	4	LWGW3 (0)	OVERLAY FOR WORK AREA 3
(270)	624	ADDRESS	4	LWCOSDN	DBKL NUMBER OF OLD CKP SLOT
(274)	628	BITSTRING	1	LWFLG2	FLAG BYTE 2
		1...		LWF2CDO	"X'80'" DELETE CKP SLOT
(275)	629	BITSTRING	3		UNUSED
(278)	632	ADDRESS	4	LW13	SAVE-AREA RD USED IN PL
(27C)	636	ADDRESS	4	LW14	SAVE-AREA RE USED IN PL
(280)	640	ADDRESS	4	LW15	SAVE-AREA RF USED IN PL
LOGICAL READER RE-DEFINITION					
(260)	608	SIGNED	4	USCC (2)	SAVE AREA FOR INSERTED RECORD
(268)	616	BITSTRING	6	LWPB (0)	SWITCH BYTES
(268)	616	BITSTRING	1	LWPI	PARAMETER ID BYTE 1
(269)	617	BITSTRING	1	LWPI2	PARAMETER ID BYTE 2
		1		PIJC	"X'01'" .. JC DOS JOB CARD READ
		.... ..1.		PIJECL	"X'02'" .. JECL MODE INDICATOR
(26A)	618	BITSTRING	1		UNUSED
(26B)	619	BITSTRING	1	LWOC	OP CODE IDENTIFIER
(26C)	620	BITSTRING	1	LWBI	PARAMETER BRANCH INDEX
(26D)	621	CHARACTER	1	LWFS	FORM SWITCH C' ' .. NORMAL RECORD C'=' .. JECL STMT REC (KEYWORD FORMAT) C',' .. JECL STMT REC (POSITIONAL FORMAT)
(26E)	622	BITSTRING	1	LWER	3540 COMMUNICATION BYTE
		1		LWERALT	"X'01'" ..CARD RDR+3540 (ALTERNATE)
		.... ..1.		LWERRD	"X'02'" ..READING FROM SECONDARY 3540, I.E. ALTERNATE OR DYNAMIC
		.... .1..		LWERDA	"X'04'" ..3540 DATA FILE PROCESSING
		.... 1...		LWERPRI	"X'08'" ..PRIMARY DISKETTE PROCESS
		...1		LWERLH	"X'10'" ..LAST DISKETTE / HOLD JOB
		..1.		LWERCB	"X'20'" ..DSHR REC CHG.SECT.BUILT BECAUSE 3540 REC LEN¬= 1ST SYSRDR
		.1..		LWERUE	"X'40'" ..UNIT EXCEPTION ON RDR
		1...		LWERDYM	"X'80'" ..DYNAMIC DISKETTE PROCESS
(26F)	623	BITSTRING	1	LWUJ	UNEXPECTED JOB INDICATOR
EXECUTION READER RE-DEFINITION					
(220)	544	BITSTRING	8		USED FOR OTHER PURPOSES
(228)	552	ADDRESS	4	TCXAAR	ADDRESS OF ACCOUNT RECORD
(22C)	556	CHARACTER	1	TCXACL	CLASS
(22D)	557	BITSTRING	1	TCXFLG	FLAG BYTE
		1		TCXRDO	"X'01'" .. READ ONLY SWITCH
(22E)	558	BITSTRING	2		UNUSED
(230)	560	ADDRESS	4	TCXPDB	ADDRESS OF PART CNTL BLOCK
(234)	564	BITSTRING	8	TCXJTD	JOB START TOD CLOCK
(23C)	572	ADDRESS	4	TCXWF0	EXEC. RDR. WORK FIELD 0
(240)	576	ADDRESS	4	TCXWF1	EXEC. RDR. WORK FIELD 1

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(244)	580	ADDRESS	4	TCXWF2	EXEC. RDR. WORK FIELD 2
(248)	584	ADDRESS	4	TCEBXR	XRE ECB DURING BAM OPEN
	EXECUTION WRITER RE-DEFINITION				
(220)	544	BITSTRING	8		USED FOR OTHER PURPOSES
(228)	552	ADDRESS	4	TCMTJH	MT PARTITION JHR POINTER
(22C)	556	ADDRESS	4	TCMTJT	MT PARTITION JTR POINTER
(230)	560	BITSTRING	16	TCLTAB	CARRIAGE CONTROL TABLE
(240)	576	CHARACTER	8	TCSECU	VSE SECURITY USERID SAVEAREA
(248)	584	CHARACTER	8	TCSECN	VSE SECURITY SECNODE " @KX40618
	EXECUTION READER AND WRITER RE-DEFINITION NOTE: OVERLAYS WORK AREA 2 AND WORK AREA 3!				
(260)	608	SIGNED	4		RESERVED FOR FUTURE USE
(264)	612	SIGNED	4	TCJGM	EX. WRITER MESSAGE ADDR.
(268)	616	SIGNED	4	TCPURC	EX. WRITER 'PURGE' RET-CODE
(26C)	620	ADDRESS	4	TCXRWA	EX. PROCESSOR WORK AREA
(270)	624	SIGNED	4	TCALET	ALET FOR PARTITION
(274)	628	BITSTRING	16	TCAAR (0)	SAVED ACC REG 1,6 - 8
(274)	628	ADDRESS	4	TCAR1	SAVED ACC REG 1
(278)	632	BITSTRING	12	TCARS (0)	SAVED ACC REG 6 - 8
(278)	632	SIGNED	4	TCAR6	SAVED ACC REG 6
(27C)	636	SIGNED	4	TCAR7	SAVED ACC REG 7
(280)	640	SIGNED	4	TCAR8	SAVED ACC REG 8
(284)	644	SIGNED	4	TCAR2	SAVED ACC REG 2
(288)	648	SIGNED	4	TCSVSP	TEMP STORAGE POINTER
(28C)	652	ADDRESS	4	TCXJNP	IPW$$XJ NEW TASK ADDR
(290)	656	ADDRESS	2	TCSVSPL	LENGTH OF TCSVSP BUF
(292)	658	ADDRESS	2	TCRRC	IPW$$XJ ERROR RETURN CODE
(294)	660	SIGNED	4	TCSR4	SAVED REG.4 (EX.WRITER)
(298)	664	ADDRESS	4	TCJGM2	COPY OF F.F. JOB GEN. MSG
(29C)	668	SIGNED	4	TCQ25ID	MSG 1Q25A ID FROM IPW$$T1
(2A0)	672	CHARACTER	7	TCXTLBL	IPW$$XJ BAM DTFNAME (LABEL)
(2A7)	679	BITSTRING	1	TCXWFG	FLAGS
		1...		TCXFTLBL	"X'80'" .. IPW$$XJ TLBL= SPEC'D
		.1..		TCXFLTPY	"X'40'" .. IPW$$XJ LTAPE=YES SPEC'D
		..1.		TCXFLTPN	"X'20'" .. IPW$$XJ LTAPE=NO SPEC'D
		...1		TCXFIPC	"X'10'" .. IPW$$XWE SAVED PG-INCRM.
(2A8)	680	SIGNED	4		UNUSED
(2AC)	684	SIGNED	4	(3)	UNUSED
	WORK AREA FOR PLOAD COMMAND PROCESSOR				
(270)	624	CHARACTER	1	TCEXTY	EXIT TYPE
		11.1 ...1		TCEXJO	"C'J'" ..JOB EXIT
		11.1 .11.		TCEXOU	"C'O'" ..OUT EXIT
		11.1 .1.1		TCEXNE	"C'N'" ..NET EXIT
		111. .111		TCEXXM	"C'X'" ..XMT EXIT
(271)	625	BITSTRING	3		RESERVED FOR FUTURE USE
(274)	628	CHARACTER	8	TCEXNA	EXIT NAME
(27C)	636	SIGNED	4	TCEXSI	EXIT SIZE
(280)	640	SIGNED	4	TCEXAD	EXIT LOAD POINT ADDRESS
(284)	644	SIGNED	4	TCEXEP	EXIT ENTRY POINT ADDRESS
		...1 1...		TCEXLE	"*-TCEXTY" ..LENGTH OF WA
	POFFLOAD TASK RE-DEFINITION				
(240)	576	CHARACTER	8	TCOONN	OLD NODE NAME
(248)	584	BITSTRING	1	TCOFLG	FLAG BYTE
		1...		TCONFT	"X'80'" .. 1ST TIME THROUGH SWITCH
		.1..		TCODKT	"X'40'" .. BUILD VSE/POWER SECTION

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
		..1.		TCOCON	"X'20'" .. CONTINUATION FLAG
		...1		TCONNN	"X'10'" .. DON'T ASSIGN NEW JOB NBR
		.... 1...		TCOADD	"X'08'" .. AT LEAST 1 Q-ENTRY ADDED
		.... .1..		TCOSEL	"X'04'" .. PERFORM SELECT FUNCTION
		.... ..1.		TCOBUP	"X'02'" .. PERFORM BACKUP FUNCTION
		1		TCOR2T	"X'01'" .. 1ST TIME THROUGH SWITCH
(249)	585	BITSTRING	1	TCOSW1	SWITCH BYTE 1
		1...		TCOSER	"X'80'" .. LAST DBLK IN DBLK GROUP
		.1..		TCOSOP	"X'40'" .. LOGICAL INTERFACE OPENED
		..1.		TCOSPI	"X'20'" .. POFFLOAD PICKUP TASK
(24A)	586	BITSTRING	1	TCORC	SAVE AREA FOR RETURN CODE
(24B)	587	BITSTRING	1		UNUSED
(24C)	588	ADDRESS	4	TCOSAL	SELECT QUEUE ARGUMENT LIST
(250)	592	CHARACTER	7	TCOTLBL	BAM DTF NAME (LABEL)
(257)	599	BITSTRING	1		UNUSED
(258)	600	SIGNED	2	TCOQRL	QRA LENGTH FOR PREVIOUS REL.
(25A)	602	SIGNED	2	TCPSHT	PICKUP TOT SCHEDULED ENTRIES
(25C)	604	SIGNED	2	TCPSAT	PICKUP TOT SAVED ENTRIES
(25E)	606	BITSTRING	8	TCPMSG	PICKUP MSG 1Q6PI TIMESTAMP
(266)	614	BITSTRING	8	TCPTMP	PICKUP TEMP WORKAREA
(26E)	622	BITSTRING	17	TCPCLA	PICKUP SAVE OF TCCT
(27F)	639	BITSTRING	1		UNUSED
	SPOOL MANAGER WORK AREA (X-PARTITION I/F)				
(240)	576	CHARACTER	8	TCJN	SPOOL MANAGEMENT JOB NAME
(248)	584	BITSTRING	2	TCJ#	SPOOL MANAGEMENT JOB NO
(24A)	586	BITSTRING	1	TCFG	FLAG BYTE COPIED FROM PIB
		1...		TCVM	"X'80'" .. VIRTUAL MODE
(24B)	587	BITSTRING	1	TCSW	SWITCH BYTE
(24C)	588	SIGNED	4	TCXA	TASK ERR EXIT RTN ADDR
	XPCC CROSS PARTITION USER TASK RE-DEFINITION				
(240)	576	ADDRESS	4	TCXTIML	TIME LIMIT
(244)	580	ADDRESS	4	TCXXPCC	ADDRESS OF XPCCB BEING USED
(248)	584	ADDRESS	4	TCXWRKA	ADDRESS OF WORKAREA
(24C)	588	ADDRESS	4	TCXEDCB	ADDRESS OF ASSOCIATED EDCB
(260)	608	ADDRESS	4	TCXCKPA	ADDR OF EXT CKP INFO
(264)	612	SIGNED	2	TCXCKPL	LENGTH OF EXT CKP INFO
(266)	614	SIGNED	2		UNUSED
(268)	616	ADDRESS	4	TCACIET	$$XTM: ADDR. OF TMP. ACIE
(26C)	620	ADDRESS	4	TCACITQ	$$XTM: ADDR. OF TQE
(1FC)	508	ADDRESS	4	TCXSPL	ADDRESS OF ASSOCIATED SPL
		1111 111.		TCXSID	"X'FE'" .. XPCC SPL PRESENT
	LOGICAL OUTPUT SPOOLER RE-DEFINITION				
(220)	544	ADDRESS	4	TCOSNR	SAVED RECORD COUNT
(224)	548	ADDRESS	4	TCOSLC	SAVED LINE/CARD COUNT
(228)	552	SIGNED	4	TCOSPC	SAVED PAGE COUNT
(22C)	556	SIGNED	2	TCOSNT	SAVED NO OF TRACKS/BLOCKS
	PRINT STATUS TASK RE-DEFINITION (QUEUE DISPLAY)				
(240)	576	ADDRESS	4	TCPSQN	NEXT QUEUE SET NUMBER
(244)	580	ADDRESS	4	TCPSWA	ADDRESS OF PS WORKAREA
(248)	584	BITSTRING	1	TCPSFG	FLAG BYTE
		1...		TCPSND	"X'80'" PROC NON-DISP CLASS CHAIN
(249)	585	CHARACTER	7	TCPSLB	BAM LABEL IF ANY
	PSTART RDR/LST/PUN TAPE TASK DEFINITION				
(240)	576	CHARACTER	7	TCTKLB	BAM LABEL IF ANY
(247)	583	BITSTRING	1		UNUSED

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
RJE,BSC TASK RE-DEFINITION					
		.111 1...		TCBQ	"TCRS" BSC APPENDAGE CHAIN PTR
		..1. 1...		TCLRQ	"TCCT+8" LCB-TO-RELEASE-QUEUE
(240)	576	ADDRESS	4	TCBS1S	LINKAGE-REG SAVE AREA
(244)	580	ADDRESS	4	TCBS2S	2ND LINKAGE-REG SAVE AREA
(248)	584	ADDRESS	4	TCBS3S	3.RD LEVEL LINKAGE SAVE
(164)	356	ADDRESS	4	TCSR	SYSREC HEADER
INITIALIZATION TASK RE-DEFINITION					
(270)	624	ADDRESS	4	TCI4RTN	RETURN ADDRESS USED BY $$AT IF $$I4 OPEN IJDTEST FAILS
(168)	360			TCEN	"*" END OF STANDARD TCB
(168)	360			TCLN	"*-TCSD" LENGTH OF TCB
EXTENSION AREA FOR 2ND DATA BLOCK BUFFER					
		If a local printer task is started with the double buffering option, the task is equipped with an expanded TCB to save specific information required. The TCB is enlarged to the next multiple of 32 bytes.			
(2B8)	696	CHARACTER	12	TC2SD (0)	
(2B8)	696	ADDRESS	4	TC2DW	2ND DATA BLOCK NUMBER
(2BC)	700	ADDRESS	4	TC2DV	VIRT ADDRESS OF 2ND BUFFER
(2C0)	704	ADDRESS	2		I/O OPERATION LENGTH
(2C2)	706	BITSTRING	1		OPERATION CODE
(2C3)	707	BITSTRING	1		RESERVED
(2C4)	708	BITSTRING	20		RESERVED FOR FUTURE
(2C4)	708			TC2LN	"(*-TCSD)" EXTENDED LENGTH OF TCB
TCB-EXPANSION FOR SAVE-ACCOUNT TASK					
		A SAVE account task is equipped with an expanded TCB to save specific information required. The TCB is enlarged to the next multiple of 32 bytes.			
(2B8)	696	CHARACTER	7	TCSAFN	TAPE/DASD FILE NAME
(2BF)	703	CHARACTER	1	TCSADY	TAPE DENSITY ... ...REFER ALSO TO TCF8MS
(2C0)	704	CHARACTER	4	TCSADV	DEVICE WHERE TO SAVE
(2C4)	708	ADDRESS	4	TCSAPB	PUB-ADDR DEV WHERE TO SAVE
(2C8)	712	ADDRESS	4	TCSAR1	DEVICE DATA PASSED FROM CP
(2CC)	716	ADDRESS	4	TCSART	LINKAGE-REG SAVE AREA
(2D0)	720	ADDRESS	4	TCSARN	2ND LINKAGE-REG SAVE AREA
(2D4)	724	ADDRESS	4	TCSADP	DTF-POINTER
(2D4)	724			TCLN1	"*-TCSD" LENGTH TCB INCL. SAVE-ACC.
		..1.		TCLN2	"*-TCSAFN" LENGTH OF EXPANSION
COMMAND PROCESSOR CONTROL FIELDS					
		THE FOLLOWING IS A RE-DEFINITION OF THE AREA FROM THE GENERAL PURPOSE WORK AREA TO THE END OF THE TCB AS USED BY THE COMMAND PROCESSOR.			
(220)	544	CHARACTER	16	CPDS	SECTION DESCRIPTOR
(230)	560	SIGNED	1	CPID	RJE-USERID (0 FOR LOCAL)
(231)	561	CHARACTER	7	CPCM	COMMAND CODE
(238)	568	CHARACTER	8	CPNO	SEQUENCE # (RJE ONLY)
(240)	576	ADDRESS	4	CPEA	ADDRESS OF CALLER ECB
(244)	580	CHARACTER	8	CPFN	FROM NODE ID
(24C)	588	CHARACTER	1	CPFQ	FROM NODE QUALIFIER (SYSID)
(24D)	589	BITSTRING	1	CPFL	FLAG BYTE FROM NMR

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
(24E)	590	CHARACTER	8	CPRT (0)	REMOTE ID | USER ID
(24E)	590	CHARACTER	4	CPR2	ORIGINATORS REMOTE ID
(252)	594	CHARACTER	4		*
(256)	598	BITSTRING	1	CPAB	AUTHORIZATION FLAGS
(257)	599	BITSTRING	1	CPFG	GENERAL FLAG BYTE
		1...		CPFP	"X'80'" ..'P'-CHAR STRIPPED OFF
		.1..		CPFC	"X'40'" ..INTERNAL CMD REQUEST
		..1.		CPFCD	"X'20'" ..CALCULATE DUE DATE
		...1		CPFSH	"X'10'" ..SHARED COMMAND VIA QCA
		.... 1...		CPPCE	"X'08'" ..PCE PASSED IN CPPA
		.... .1..		CPVER	"X'04'" ..VERIFIED DCLT IN CPPA
		.... ..1.		CPNLK	"X'02'" ..DO NOT LOCK DPCB
		1		CPDCK	"X'01'" ..DELETE CKP INFO
(258)	600	BITSTRING	1	CPFG2	GENERAL FLAG BYTE 2
		1...		CP2QN	"X'80'" ..USE DIRECT Q-REC NUMBER
		.1..		CP2PA	"X'40'" ..POST ATTENTION ROUTINE
		..1.		CP2BU	"X'20'" ..POWER CMD PROCESSOR BUSY
		...1		CP2OA	"X'10'" ..CMD HAS OPERATOR AUTHORITY
		.... 1...		CP2SE	"X'08'" ..ENTRY NOT SPOOL ACC PROT"D@KXD0337
(259)	601		2	CPO#	CURRENT OPERAND NUMBER
(25B)	603	BITSTRING	1	CPRL	REPLY LENGTH
(25C)	604	CHARACTER	72	CPOP	OPERANDS (FREE FORMAT)
(2A4)	676	CHARACTER	58		EXTRA OPERANDS
		1... ..1.		CPOPL	"*-CPOP" NEW OPERAND LENGTH
(2DE)	734	CHARACTER	2		RESERVED
(2E0)	736	CHARACTER	8	CPPW	PASSWORD OF ISSUER
(2E8)	744	ADDRESS	4	CPOR	OWNER OF REQUEST (0 FOR AR ROUTINE COMMAND PROCESSOR)
(2EC)	748	CHARACTER	8	CPXA	XPCC APPLICATION ID
(2F4)	756	SIGNED	4	CPPA	'PASS' VALUE FROM IPW$ICP
(2F8)	760	BITSTRING	2	CPTIK	AR ROUTINE TIK
(2FA)	762	CHARACTER	2		RESERVED FOR FUTURE USE
(2FC)	764	CHARACTER	8	CPCON	AR CONSOLE NAME
(304)	772	CHARACTER	8	CPSUS	USERID FROM PWRSPL/SPL
(30C)	780	CHARACTER	8	CPRTU	USERID FOR AUTHORIZATION CK
(314)	788	CHARACTER	36		RESERVED FOR FUT @KXC0192
(314)	788			CPLN	"(*-CPDS)" LENGTH OF CMND CNTRL FIELDS
(314)	788			TCCL	"*-TCSD" LENGTH OF EXTENDED TCB AREA
	OVERLAY FOR PNET TASKS				
(220)	544	ADDRESS	4	TCENCB	ADDRESS OF NODE CTRL BLOCK
(224)	548	ADDRESS	4	TCENTE	ADDR OF NCB TASK ENTRY
(228)	552	BITSTRING	1	TCERCB	RCB OF TASK CONCERNED
(229)	553	BITSTRING	1	TCETTC	TERMINATION CONDITION BYTE
		1...		TCETSO	"X'80'" .. SIGNOFF RECORD SENT/REC
		.1..		TCETLC	"X'40'" .. LINE ERROR STOP
(22A)	554	BITSTRING	2	TCEFCS	FCS BYTES
(22C)	556	ADDRESS	4	TCEWKA	ADDRESS OF WORKAREA
(230)	560	BITSTRING	1	TCEST1	STATUS BYTE 1
		1...		TCERIF	"X'80'" .. RIF SENT/RECEIVED
		.1..		TCEPGR	"X'40'" .. PERMISSION GRANTED SENT/RECEIVED
		..1.		TCEPRJ	"X'20'" .. PERMISSION REJECTED SENT/RECEIVED

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
		..1.		TCERCS	"X'20'" .. RECEIVER CANCEL SENT/RECEIVED
		...1		TCEEOF	"X'10'" .. EOF SENT/RECEIVED
		.... 1...		TCEADS	"X'08'" .. ABORT TRANSMISSION SENT/RECEIVED
		.... .1..		TCECMC	"X'04'" .. TRANSMISSION COMPLETE SENT/RECEIV
(231)	561	BITSTRING	1	TCEST2	STATUS BYTE 2
		1...		TCEWIB	"X'80'" .. WAITING FOR INPUT BUFFER
		.1..		TCECRTL	"X'40'" .. COMPRESSION ERROR
		..1.		TCENOP	"X'20'" .. DON'T POST THIS TASK
		...1		TCESPD	"X'10'" .. TASK SUSPENDED
		.... 1...		TCEPBO	"X'08'" .. POST ONLY AFTER BUFFER SENT
		.... .1..		TCERCA	"X'04'" .. RECEIVER CANCEL AFTER ABORT SENT
		.... ..1.		TCERAB	"X'02'" .. RELEASE OF ALL BUFFERS REQUESTED
		1		TCESOB	"X'01'" .. SHORT ON BUFFER CONDITION
(232)	562	BITSTRING	2		UNUSED
PART FOR RECEIVER TASK					
(234)	564	ADDRESS	4	TCERHD	ADDR OF RECEIVED INPUT BUFFER QUEUE
(238)	568	ADDRESS	4	TCERTL	TAIL PTR RECEIVED INPUT BUFFER QUEUE
(23C)	572	BITSTRING	1	TCENRB	NUMBER OF RECEIVED BUFFERS
(23D)	573	BITSTRING	3		UNUSED
(23D)	573			TCELN	"*-TCSD" LENGTH EXTENDED TCB FOR PNET
OVERLAY FOR TRANSMITTER TASK					
(234)	564	ADDRESS	4	TCEFOB	ADDR OF FREE OUTPUT BUFFER QUEUE
(238)	568	BITSTRING	1	TCENAB	NUMBER OF ACQUIRED BUFFERS
(239)	569	SIGNED	3	TCETL#	TOTAL LINE NUMBER
(23C)	572	SIGNED	4	TCECL#	CURRENT LINE NUMBER
SNA MANAGER CONTROL FIELDS					
THE FOLLOWING IS A RE-DEFINITION OF THE GENERAL WORK AREA USED BY THE SNA MANAGER. THE FOLLOWING LIST CONTAINS THE ECB ADDRESSES ON WHICH THE SNA MANAGER ISSUES A MULTIPLE WAIT.					
(220)	544	CHARACTER	12	TCEL (0)	WAIT ECB LIST
(220)	544	ADDRESS	4	TCE1	RECEIVE ANY ECB ADDR
(224)	548	ADDRESS	4	TCE2	WORK ECB ADDR
(228)	552	BITSTRING	1	TCED	END OF LIST
(229)	553	CHARACTER	3		NOT USED
(22C)	556	SIGNED	4		RESERVED
(240)	576	ADDRESS	4	TCLU	ADDRESS OF LUCB
(244)	580	ADDRESS	4	TCWA	ADDRESS OF WORK AREA
(248)	584	ADDRESS	4	TC13	SAVE AREA FOR R13
(24C)	588	ADDRESS	4	TCRPL	ADDRESS OF RECEIVE ANY RPL
OUTBOUND (OB) PROCESSOR CONTROL FIELDS					
THE FOLLOWING IS A RE-DEFINITION OF A PART OF THE GENERAL WORK AREA FOR THE OUTBOUND PROCESSOR.					
(24C)	588	SIGNED	4	TCRO	ECB FOR COMMANDS (E.G SETUP)
(24C)	588			TCROEL	"TCRO+2" REACT OB EVENT LIVE BYTE
		1...		TCROL	"X'80'" .. REACT OB LIVE BIT

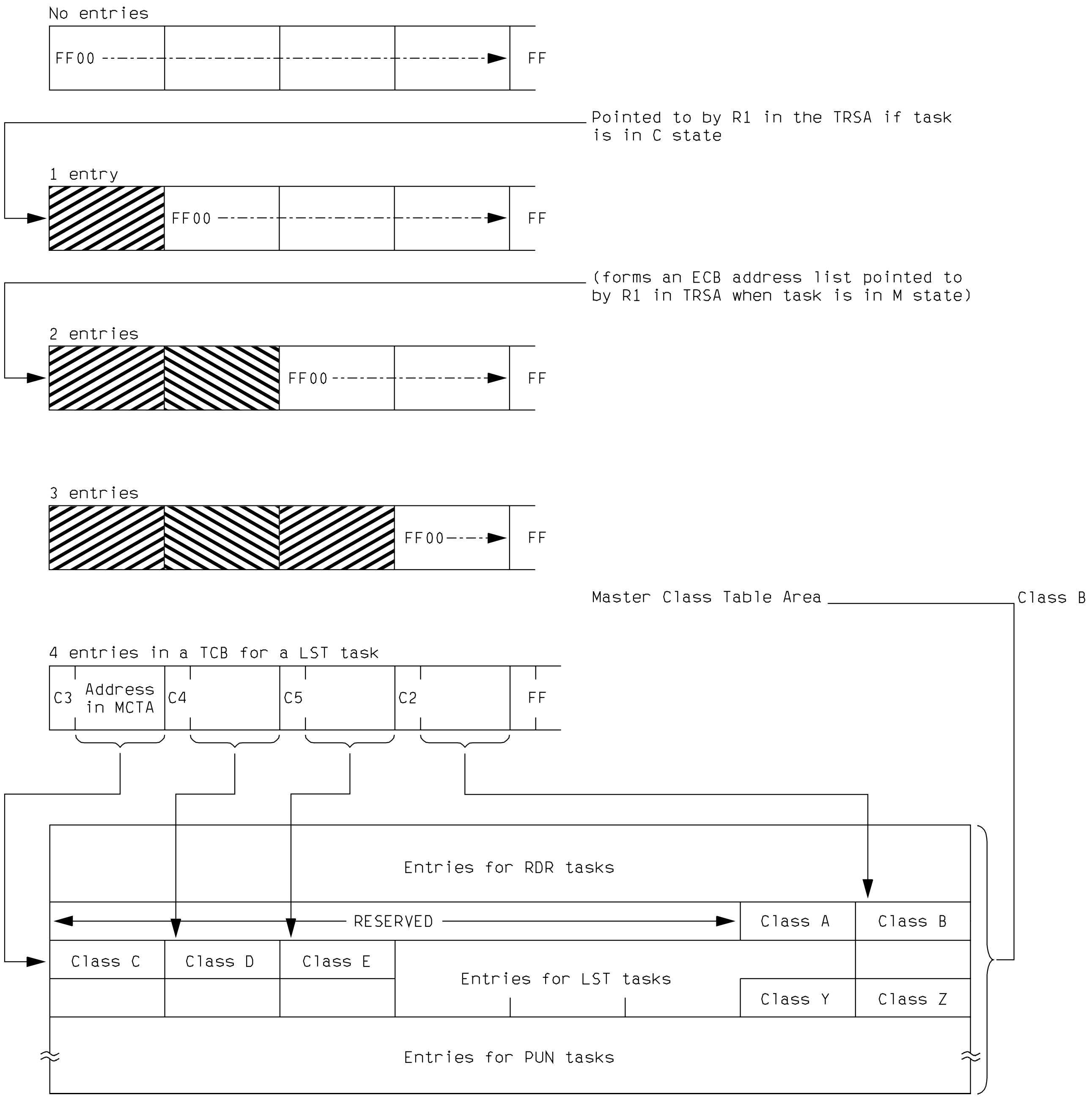


Figure 145. Task Class List

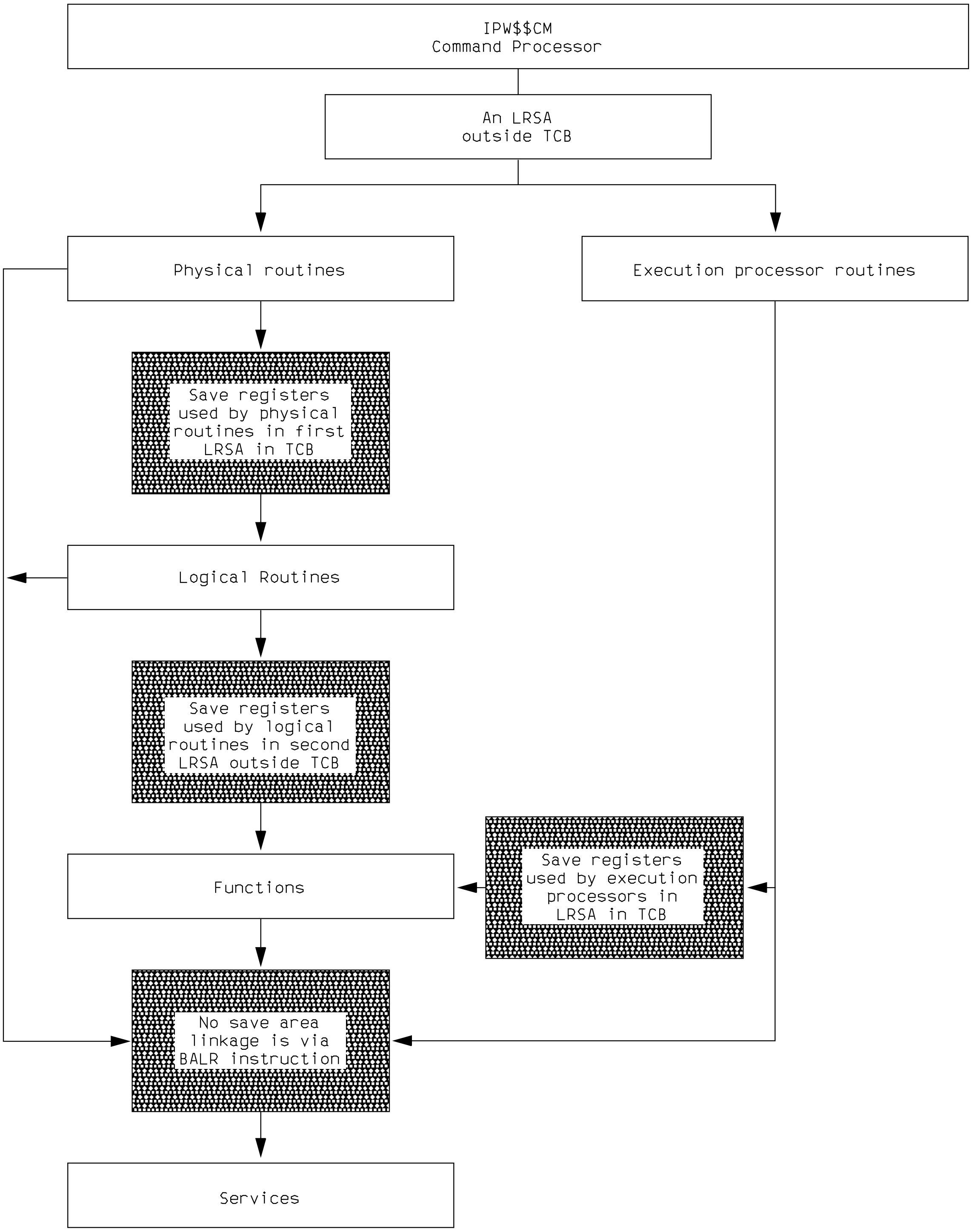


Figure 146. Summary of Linkage Register Save Areas

Command Processor Control Block (CPB)

This block replaces part of a command processor TCB, when a command is entered via the console keyboard by the central operator, and of its associated temporary command processor TCB when linkage is made via the IPW$ICP macro.

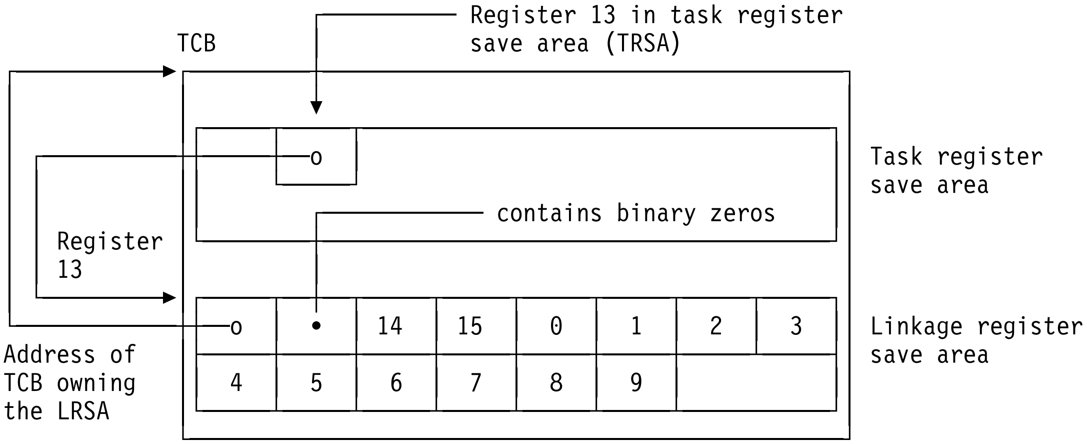
Definition Macro: IPW$DTC CP=YES

The CPB replaces the general task work area of standard TCB. The contents of the CPB are described in the Task Control Block (TCB) "Command Processor Control Fields" (see “Task Control Block (TCB)” on page 690).

Additional Linkage Register Save Area (LRSA)

Included by definition macro IPW$DSV for the save area.

The linkage register save area of the TCB is required by each routine in order to save the registers if the routine needs a function. See Figure 147.



Register 13 in task register save area (TRSA)
TCB
o
contains binary zeros
Task register save area
Register 13
o
•
14
15
0
1
2
3
4
5
6
7
8
9
Linkage register save area
Address of TCB owning the LRSA

Figure 147. Linkage from a Physical Routine to a Function Routine

An additional linkage register save area is required by some tasks to link routines within the tasks; in particular this is necessary when one function routine invokes another function routine. This LRSA has the same format as the LRSA described in the TCB. A new linkage register save area is built by acquiring storage for the save area by means of the IPW$RSW macro instruction and chaining the new save area, whereby making the current save area to the previous and the new save area to the current one. Register 13 points always to the current save area. The first fullword of the save area is initialized to address the TCB of the issuing task. The second fullword of the save area is initialized to address the previous save area.

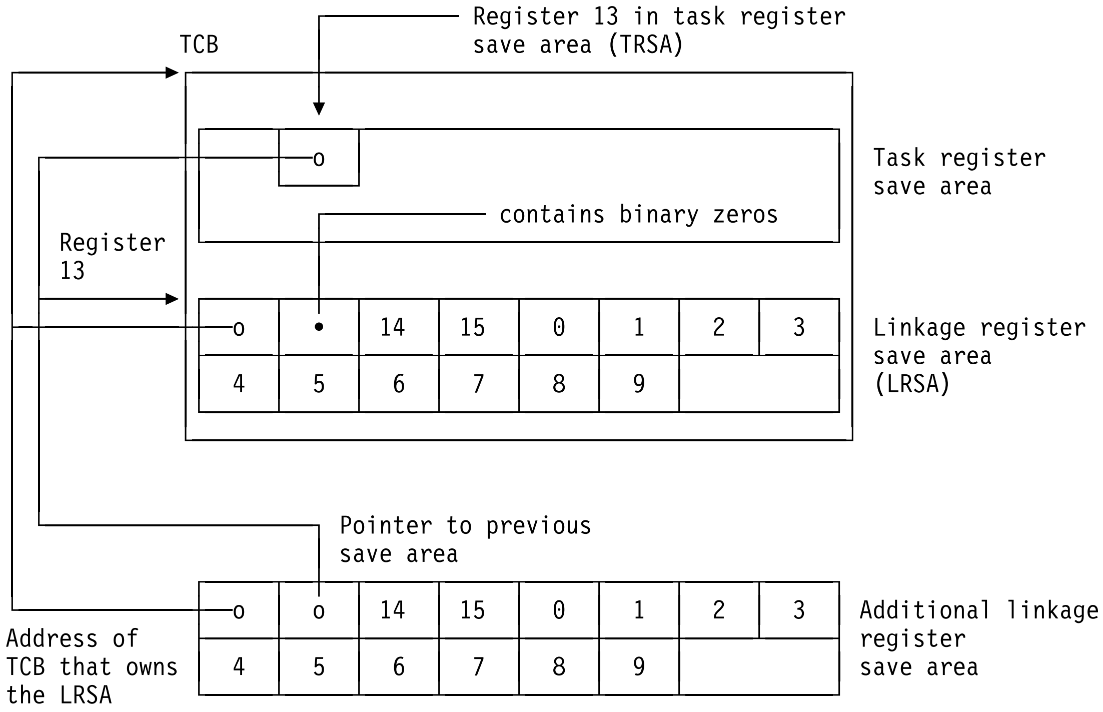

Register 13 in task register save area (TRSA)
TCB
o
Task register save area
contains binary zeros
Register 13
o
•
14
15
0
1
2
3
4
5
6
7
8
9
Linkage register save area (LRSA)
Pointer to previous save area
o
o
14
15
0
1
2
3
4
5
6
7
8
9
Additional linkage register save area
Address of TCB that owns the LRSA

Figure 148. Linkage from One Function Routine to Another Function Routine

Linkage from a Physical Routine to a Logical Routine: Execution of the IPW$OLI macro instruction causes the creation of a second LRSA. The first LRSA is associated with the physical routine issuing the macro instruction (physical save), and the second LRSA is associated with the logical routine invoked by the macro instruction (logical save). The linkage register save areas are double-threaded. The first fullword of the save area associated with the physical routine contains the address of the save area associated with the logical routine. The second fullword contains the address of any previous save area. The first fullword of the save area associated with the logical routine contains the address of the TCB of the issuing task while the second fullword addresses the previous save area. The address of the logical routine entry point is stored in the third word of the linkage register save area.

This is referred to as double linkage register save area (DLRSA). Linkage between the two LRSAs in a DLRSA is shown in Figure 149 on page 717 and Figure 150 on page 718.

Double Linkage Register Save Area (DLRSA): Case 1, where the task is executing in the physical routines (PR, PL, PP), is shown in Figure 149.

Case 2, where the task is executing in the logical routines (LR, LW), is shown in Figure 150 on page 718.

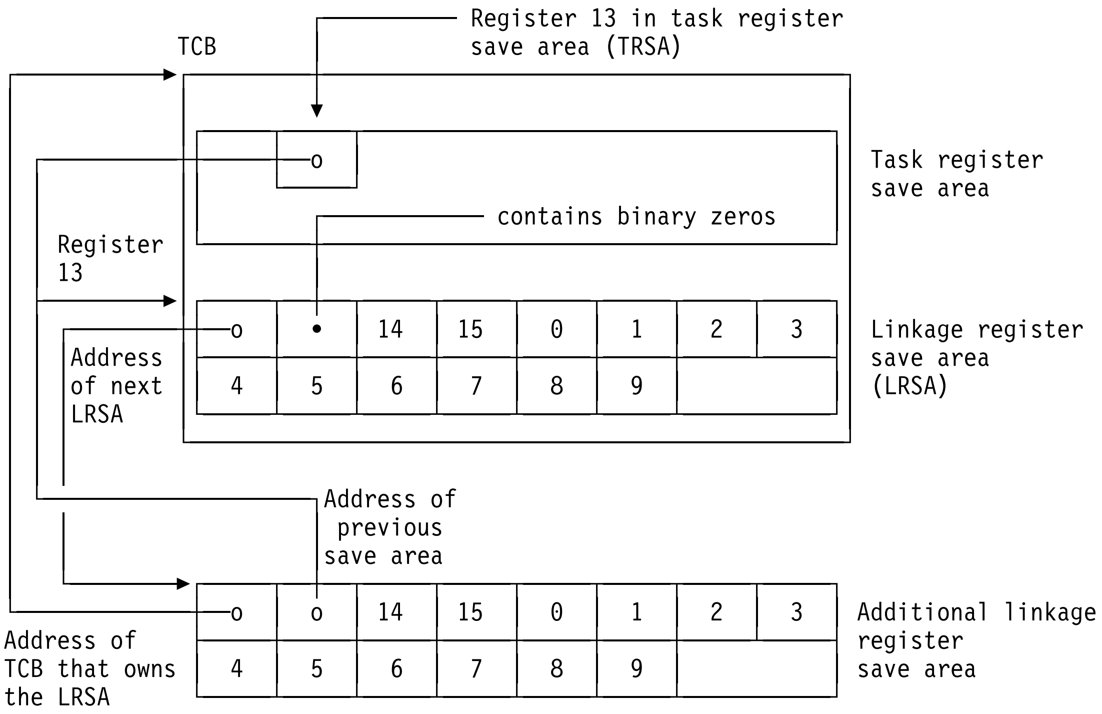

Register 13 in task register save area (TRSA)
TCB
o
Task register save area
contains binary zeros
Register 13
o
•
14
15
0
1
2
3
4
5
6
7
8
9
Linkage register save area (LRSA)
Address of next LRSA
Address of previous save area
o
o
14
15
0
1
2
3
4
5
6
7
8
9
Additional linkage register save area
Address of TCB that owns the LRSA

Figure 149. Linkage Between the Two LRSAs in a Double Linkage Register Save Area (Case 1)

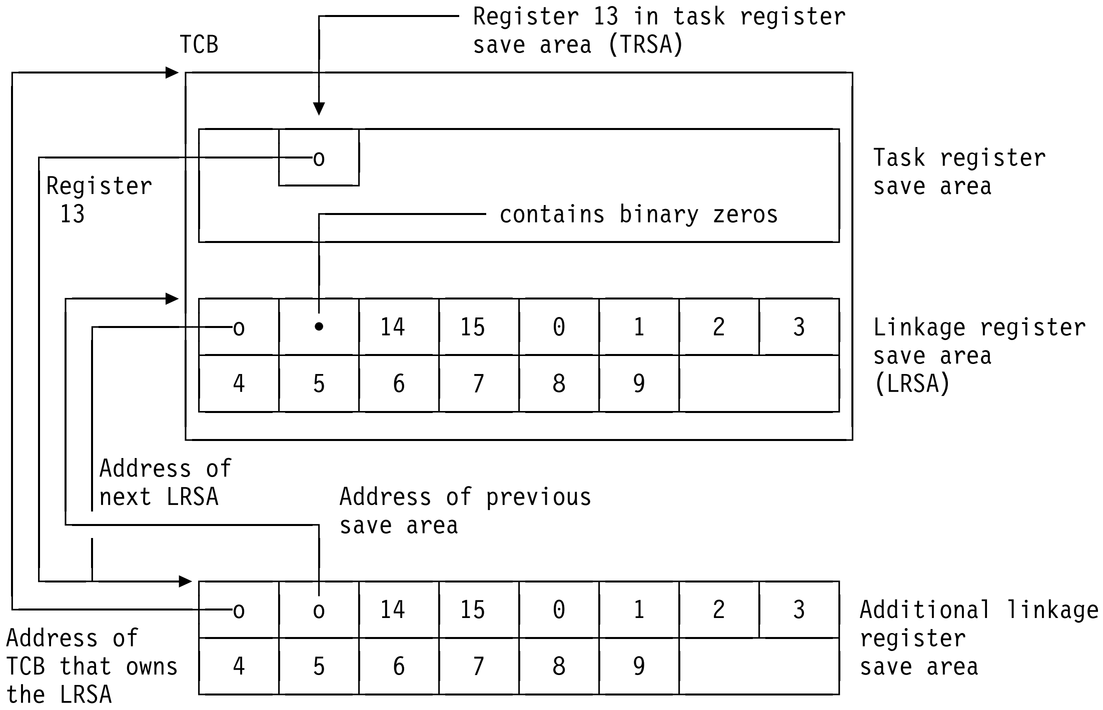

Register 13 in task register save area (TRSA)
TCB
Register 13
o
Task register save area
contains binary zeros
o
•
14
15
0
1
2
3
4
5
6
7
8
9
Linkage register save area (LRSA)
Address of next LRSA
Address of previous save area
o
o
14
15
0
1
2
3
4
5
6
7
8
9
Additional linkage register save area
Address of TCB that owns the LRSA

Figure 150. Linkage Between the Two LRSAs in a Double Linkage Register Save Area (Case 2)

Execution of the IPW$CLI macro instruction causes destruction of the interface linkage previously established by the IPW$OLI macro instruction and release of the additional LRSA acquired by that instruction. Once the IPW$CLI macro instruction has been issued no further IPW$GLR or IPW$PLR macro instructions may be issued until the next IPW$OLI macro instruction is issued.

Linkage from a Physical/Logical Routine to a Function: 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 table (CAT).

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.

Immediately upon entry the contents of registers 14 through 9 are saved in words 3 through 14 of the LRSA provided by the calling routine and addressed by register 13.

On return from a function, registers 14 through 9 are restored from the LRSA addressed by register 13. A branch is then made to the return address now contained in register 14.

Linkage to a Service: No registers are saved, other than in the TCB, when going from one of the following:

- External routine to a service.
- Internal routine to a service.
- Function to a service, except in the case of calling storage management, when registers 14 through 7 are stored in the SCB, and in the case of calling message service when register 5 is stored in the MMB.

Note: Any service may use registers 0 through 3 destructively.

Task Dispatch Trace Area

Definition Macro: IPW$DEF TTRACE=YES

The Task Dispatch Trace area resides in real storage and shows history about the last 'n' dispatched tasks and what the task's status at dispatch time was. Each trace entry is X'80' bytes long.

- Task Dispatch Trace Area Header

Bytes Hex.	Label of Field	Description/Function
00-0F	TTRHSD	Storage descriptor
10-13	TTRHBEG	Address of first entry
14-17	TTRHEND	Address of last entry
18-1B	TTRHUSE	Address of last used entry
1C-1F	TTRHSIZ	Total size of trace area header
	TTRHLEN	Length of trace area header

- Task Dispatch Trace Area Entry

Bytes Hex.	Label of Field	Description/Function
00-07	TTRTID	Task identifier and cuu
08-0B	TTRADDR	Address of TCB of task
0C	TTRFT	Task function trace indication
0D	TTRTT	Task termination byte
0E	TTRSFB	First byte of task selection field
0F		Reserved for future use
10-18	TTRF0210	Task flag bytes 2-10
19-1F	TTRINT	Task interface and function request bytes
20-57	TTRTR	Task registers 12-9
58-5F	TTRSTCK	Time stamp in STCK format
60-63	TTRTRAC1	Task Access Register 1
64-6F	TTRTRAC6	Task Access Registers 6, 7 and 8
70-76	TTRF1117	Task Flag Bytes 11-17
77-7F		Reserved for future use
	TTRLENG	Length of trace entry

Timer Queue Element (TQE)

This control block is used to control timer intervals set up by a VSE/POWER task. One timer queue element (TQE) exists for each interval currently setup.

Definition Macro: IPW$DEF TQE=YES

Bytes	Label	
Hex.	of Field	Description/Function
00-03	TQENE	Address of next TQE in chain
04-0B	TQETI	Interval end time (TOD units)
0C-0F	TQEECBP	ECB or address of ECB
10-13	TQEOTCB	Requestor's TCB address
14	TQEFLG	Flag byte
	TQEECB	X'80' - ECB within timer queue element
	TQECAN	X'40' - Cancel of TQE requested
	TQEACT	X'20' - TQE active
15-17		Reserved for future use

Trace Information Block (TIB)

This control block is used to control the internal trace facilities of RJE/BSC and PNET if the TRACE function has been requested on the PSTART of the line or node.

Definition Macro: IPW$DEF TIB=YES

Offset Hex	Offset Dec	Type	Len	Name (Dim)	Description
				TRACE INFORMATION BLOCK (TIB)	
(0)	0	CHARACTER	16	TIBSD	SECTION DESCRIPTOR
(10)	16	BITSTRING	1	TIBFLG1	TIB FLAGBYTE 1
		1...		TIBDSB	"X'80'" .. DUMP SUBTASK BUSY
(11)	17	BITSTRING	3		UNUSED
(14)	20	ADDRESS	4	TIBPTRC	POINTER TO AREA IN USE
(18)	24	BITSTRING	4		RESERVED
(1C)	28	SIGNED	4	TIBLCK	LOCK-WORD
(20)	32	CHARACTER	48	TIBRSA (0)	REGISTER SAVE AREA
(20)	32	SIGNED	4	TIBRE	REGISTER 14
(24)	36	SIGNED	4	TIBRF	REGISTER 15
(28)	40	SIGNED	4	TIBR0	REGISTER 0
(2C)	44	SIGNED	4	TIBR1	REGISTER 1
(30)	48	SIGNED	4	TIBR2	REGISTER 2
(34)	52	SIGNED	4	TIBR3	REGISTER 3
(38)	56	SIGNED	4	TIBR4	REGISTER 4
(3C)	60	SIGNED	4	TIBR5	REGISTER 5
(40)	64	SIGNED	4	TIBR6	REGISTER 6
(44)	68	SIGNED	4	TIBR7	REGISTER 7
(48)	72	SIGNED	4	TIBR8	REGISTER 8
(4C)	76	SIGNED	4	TIBR9	REGISTER 9
(50)	80	ADDRESS	4	TIBCFTE	CURRENT FREE TRACE ENTRY
	THE FOLLOWING TWO FIELDS ADDRESS THE PRIMARY TRACE AREA				
(54)	84	ADDRESS	4	TIBPTRB	PRIME TRACE AREA BEGIN
(58)	88	ADDRESS	4	TIBPTRE	PRIME TRACE AREA END
	THE FOLLOWING TWO FIELDS ADDRESS THE ALTERNATE TRACE AREA				
(5C)	92	ADDRESS	4	TIBATRB	ALTERNATE TRACE AREA BEGIN
(60)	96	ADDRESS	4	TIBATRE	ALTERNATE TRACE AREA END
(64)	100	BITSTRING	24	TIBSRB	FOR SERVICE REQUEST BLOCK
	LINKAGE REGISTER SAVE AREA				
			THE FOLLOWING AREA CONTAINS THE POINTER TO THE PREVIOUS SAVE AREA WHENEVER IPW$$AS IS CALLED.		
(7C)	124	CHARACTER	56	TIBSV	REGISTER SAVE AREA
(B4)	180	ADDRESS	1		
		1.11 1...		TIBLN	"*-TIBDS" LENGTH OF CONTROL BLOCK

User Exit Data Table

Definition Macro: IPW$DEF EXTAB=YES

The user exit table is anchored in VSE/POWER's Control Address Table in field CAEXTAB. The table contains data about the currently loaded VSE/POWER user exits. For each single exit a table entry exists. An empty table entry is flagged with X'00' in the first byte of an entry. EXTBNUM specifies the maximum number of table entries including the logical end indicator X'FF'.

Offset Hex	Type	Len	Name (Dim)	Description
(0)	CHARACTER	1	EXTYPE	EXIT TYPE
	CHARACTER		EXJOB	"C'J'" ..JOB (RDR) EXIT
	CHARACTER		EXOUT	"C'O'" ..OUTPUT EXIT
	CHARACTER		EXNET	"C'N'" ..PNET RECEIVER EXIT
	CHARACTER		EXXMT	"C'X'" ..PNET TRANSMITER EXIT
	1111 1111		EXTBEND	"X'FF'" ..LOGICAL END INDICATOR
	....		EXEMPTY	"X'00'" ..EMPTY ENTRY INDICATOR
(1)	CHARACTER	1	EXSTAT	EXIT STATUS
	CHARACTER		EXENAB	"C'E'" ..EXIT ENABLED
	CHARACTER		EXDISAB	"C'D'" ..EXIT DISABLED
	CHARACTER		EXFAIL	"C'F'" ..EXIT FAILED
(2)	BITSTRING	1		EXIT ENTRY FLAG BYTE 1
	1...		EXF1PU	"X'80'" .. RUN AS PARALLEL WORKUNIT
(3)	CHARACTER	1		RESERVED FOR FUTURE USE
(4)	CHARACTER	8	EXNAME	EXIT NAME
(C)	SIGNED	4	EXADDR	EXIT ADDRESS
(10)	SIGNED	4	EXSIZE	LENGTH OF EXIT
(14)	ADDRESS	2	EXWAL	LENGTH OF EXIT WORK AREA
(16)	SIGNED	2		RESERVED FOR FUTURE USE
(18)	SIGNED	4	EXEPAD	EXIT ENTRY POINT
(1C)	SIGNED	4		RESERVED FOR FUTURE USE
	.... .1.1		EXTBNUM	"5" NUMBER OF TABLE ENTRIES
	..1.		EXTBELN	"*-EXTYPE" LENGTH OF A TABLE ENTRY
	1.1.		EXTBLN	"EXTBELN*EXTBNUM" LENGTH OF TABLE

How to Locate: Refer to Figure 151 on page 730 in Chapter 6, “Diagnostic Aids.”

Virtual Buffer Control Area (Prefix)

Definition Macro: IPW$DBA

When virtual storage (GETVIS) is required by a VSE/POWER task, storage management precedes each storage area with a buffer control area, which indicates to which pool the storage area belongs.

Bytes	Label	
Hex.	of Field	Description/Function
00-03	BCABL	Length of storage area reserved
04	BCAPID	Pool identifier
05-07	BCATCB	TCB address of owning task
08-0B	BCAFWD	Pointer to next storage element
0C-0F	BCABWD	Pointer to previous storage element

VTAM Driver Control Block (VDCB)

Definition Macro: IPW$DVC

This control block is created by the command processor when the first connection to a node using SNA is started. Its address can be found in the PNCB at label 'PNCBVDCB'.

Bytes	Label	
Hex.	of Field	Description/Function
00-0F	VDCBSD	Storage descriptor
10-13	VDCBECB	PNET SNA event control block (ECB)
14-17	VDCBPSRQ	Pointer to SRQE chain
18-1B	VDCBPLDQ	Pointer to parked SRQEs by PNET driver
1C-33	VDCBTIME	Timer queue element

- Activity / Communication / Status Bytes

Bytes Hex.	Label of Field	Description/Function
34-35	VDCBASES	Number of active sessions
36	VDCBACT1	Activity flag byte 1
	VDCBSSUP	X'80' - SNA start up request
	VDCBOPRT	X'20' - Successful SNA open done
	VDCBSSTP	X'10' - SNA stop request set by TPEND
	VDCBCON	X'08' - Connect request
	VDCBPEND	X'04' - PEND given by operator
37	VDCBACT2	Activity flag byte 2
38	VDCBCOMS	Communication byte
	VDCBLGQ	X'80' - SETLOGON quiesce request
	VDCBSCL	X'40' - VTAM close request
	VDCBMSG	X'01' - Suppress warning message 1RE0I
39	VDCBSTA1	SNA intertask status byte
	VDCBOPEN	X'80' - SNA open successful
	VDCBOPNF	X'40' - SNA open failed
	VDCBLOGF	X'20' - SNA setlogon failed
	VDCBOPNP	X'04' - SNA open pending
	VDCBNPWR	X'02' - NO VSE/POWER termination request allowed
3A	VDCBTTC	SNA termination code
	VDCBTTCV	X'80' - VTAM abend or HALT quick
	VDCBTTCE	X'20' - Normal shutdown request
3B	VDCBTTCQ	Termination code qualifier
	VDCBVEOJ	X'00' - VTAM HALT NET
	VDCBHALT	X'04' - VTAM HALT IMM (HALT QUICK)
	VDCBVTAB	X'AB' - VTAM abend
3C	VDCBRCNT	Retry counter
3D-3F		Reserved for future use

- The following part is used for the VTAM ACB:

Bytes Hex.	Label of Field	Description/Function
40-77	VDCBACB	ACB for VTAM

Virtual Storage Control Block (VSCB)

The virtual storage control block is used to control access to the virtual storage management routines.

Definition Macro: IPW$DVS

Bytes	Label	
Hex.	of Field	Description/Function
00-0F	VSSD	Section descriptor (VSB .....)
10-13	VSAN	Anchor for system queue
14-17		Tail pointer for system queue
18-1B	VSEB	ECB
1C-1F	VSLK	Lockword
20-4F	VSTR	Register save area
50-53	VSMAX	Maximum number of bytes allocated
54-57	VSCUR	Current number of bytes allocated
• Subpool Section		
58-5D	VSGN	General pool (IPWGEN)
5E-5F		Pool id assigned by VSE/AF
60-65	VSMG	Message/command pool (IPWMSG)
66-67		Pool id assigned by VSE/AF
68-6D	VSPN	Network pool (IPWNET)
6E-6F		Pool id assigned by VSE/AF
70-75	VSSNA	RJE, SNA pool (IPWSNA)
76-77		Pool id assigned by VSE/AF
78-7D	VSSNA2	RJE, SNA WACB + COCB pool (IPWWAC)
7E-7F		Pool id assigned by VSE/AF
80-A8		Reserved for GETVIS counts

Wait Control Block (WCB)

Definition Macro: (None - located in IPW$$NU)

The wait control block is a skeleton task control block used to delimit the task selection list. The wait control block occupies locations in the permanent area of the VSE/POWER partition. The format of the wait control is as follows.

Bytes Hex.	Label of Field	Description/Function
00-0F	TMSD	Storage descriptor (WCB)
10-13		Reserved
14-17	TMTN	Address of TCB belonging to task with highest priority in TSL
18-1B	TMPF	Page fault request word - always zero
1C-1F	TMSF	Task selection field
1D - 1F		Address of routine that tests if a VSE/POWER event is posted in main ECB. If not, it places the VSE/POWER partition in wait state by issuing an SVC7.

How to Locate: Refer to Figure 151 on page 730 in Chapter 6, “Diagnostic Aids.”

Chapter 6. Diagnostic Aids

This section consists of hints and suggestions about where and what to look for in a dump containing the VSE/POWER partition and the SVA part of VSE/POWER. The section begins with general debugging hints, a list of which follows.

- The stand-alone dump (DOSVSDMP)
- Identifying the VSE/POWER partition (the partition in which VSE/POWER is initialized)
- Identifying the SVA part of VSE/POWER
- Identifying pages belonging to the fixable area
- Identifying the start of the pageable area
- Locating and identifying control blocks, tables and areas
- Identifying the start of a CSECT
- Establishing the "level" of a CSECT
- Determining the active routine and analyzing the register save areas.
- Analyzing event control blocks
- Using the buffer control words
- Analyzing TCBs
- RJE,BSC and PNET trace facility
- PNET BSC I/O logging on console
- VSE/POWER file dump program
- Establishing the last command issued
- An aid to eliminate components
- Problems related to VTAM.

General Debugging Hints

Stand-alone Dump

It is recommended that the user generates a stand-alone dump tape using DOSVSDMP.

This dump should always be used when a stand-alone dump is required. Later printing may be done for total dump tapes using DOSVSDMP or for selected area with the INFO/ANALYSIS tool.

Identifying the VSE/POWER Partition

The start of the VSE/POWER partition can be easily identified in the translated portion of any dump by the name given to the POWER macro. A copyright statement with product number 5686-066 follows at offset X'80'.

Identifying the SVA Part of VSE/POWER

The characters CAT, followed by the copyright statement with product number 5686-066, identify the control address table in the SVA and so the beginning of the SVA part of VSE/POWER.

Identifying Fixed Pages

The address of the first page in the fixable area is contained in bytes X'48-4B' (PAFA) of the SVA part of VSE/POWER. Since each page is 4K bytes, the start of other pages in the fixable area can be calculated. Also, by following the BCW chain and examining the contents of the buffer control words the amount of pages and usage of each page can be established.

Identifying the Start of the Pageable Area

The address of the pageable area is contained in bytes X'4C-4F' (PAVA) of the SVA part of VSE/POWER.

Locating and Identifying Control Blocks, Tables, and Areas

In the SVA Part Control blocks, tables, and areas in the SVA part area can be found by reference to Figure 151.

Figure 151. Locating and Identifying Control Blocks, Tables and Areas in the SVA Part

Abbreviated * Mnemonic of Table or Area	Pointer to or Address of the Table/Area	Identifier in Translated Dump
CAT	X'5C'(IJBPWR) of SYSCOM or X'14' of VSE/POWER partition	CAT and ver/mod level
Real storage control block	X'114'(CASC) of CAT	SCB
Local message control blk.	X'118'(CAMM) of CAT	MMB
Wait control block (WCB)	X'214'(TATM) of CAT	WCB
Partition control block	X'A0'(POWPCB) of partition COMREG X'230'(TCXPDB) of EX RDR TCB	PART.CONTR.BLOCK

* Refer to “List of Abbreviations” on page 799.

In the Permanent Area: There are no control blocks in the permanent area. The permanent area consists of the RJE/BSC manager if RJE is generated. If not, no permanent area exists. The page belongs then to the fixable area.

In the Fixable Area: Control blocks, tables, and areas in the fixable area can be found in Figure 152 on page 731. Actual tables present depend on task requirements.

Figure 152 (Page 1 of 3). Locating and Identifying Control Blocks, Tables, and Areas in the Fixable Area

Abbreviated * Mnemonic of Table or Area	Pointer to or Address of the Table/Area	Identifier in Translated Dump
Disk management block DMB	X'10C' of CAT (CAQC)	DMB
Master class table (MLTA)	X'39C' - X'723' of DMB (QCCT)	
Master line table (MLT)	X'78' - X'87' of DMB (MRLT)	
SYSID class table	X'8E0' - X'96F' of DMB (SSST)	
Node attached table (NAT)	X'970' - X'C8F' of DMB (MNAT)	
Master record area (MRA)	X'300' - X'FE7' of DMB (fixed part)(QCMR) X'FE8' start of variable part	
Queue control area info	X'C9A' - X'CCF' of DMB (QCASNGP)	
Relative DBLK number of current slot	X'44'-X'47' of DMB (QCADW)	
Relative DBLK number of first slot	X'CA0-X'CA3' of DMB (QCASDSA)	
Virtual addr of slot DBLK	X'48'-X'4B' of DMB (QCADV)	
Account file seek address last record	X'388' - X'38F' of DMB (MRAS)	
Account control block	X'110' of CAT (CAAC)	
Auxiliary Queue record address	X'30' of DMB (QCQW+4)	
Perm cmd processor TCB	X'218' of CAT (TAOC)	TCB O CP
Cmd proc. control fields	X'220' of the CP TCB (TCGW)	CPB
End address of VSE/POWER partition	X'50' of CAT (PAEN)	
First fixed page	X'10' of SCB (SCFP)	
INIT/TERM TCB	X'21C' of CAT (TAIT)	TCBbIbIT
Address of first LCB	X'52C' of CAT (CALC)	
Logical data area LDA	X'168' of a TCB (TCDV) (only if applicable)	Virtual address
LRSA (linkage reg save area)	X'94' - X'CB' of a TCB for an RDR, LST, PUN, or XP task (TCSV)	This LRSA saves R14-R9 used by the physical routines
MCB for Q file	X'150' of CAT (CAQ1)	MCB QFILE 01
MCB data file 1	X'154 of CAT (CAD2)	MCB DFILE 02
MCB data file 2	X'158' of CAT (CAD3)	MCB DFILE 03
MCB data file 3	X'15C' of CAT (CAD4)	MCB DFILE 04
MCB data file 4	X'160' of CAT (CAD5)	MCB DFILE 05
MCB data file 5	X'164' of CAT (CAD6)	MCB DFILE 06
MCB data file 6	X'168' of CAT (CAD7)	MCB DFILE 07
MCB data file 7	X'16C' of CAT (CAD8)	MCB DFILE 08
MCB data file 8	X'170' of CAT (CAD9)	MCB DFILE 09
MCB data file 9	X'174' of CAT (CAD10)	MCB DFILE 10
MCB data file 10	X'178' of CAT (CAD11)	MCB DFILE 11
MCB data file 11	X'17C' of CAT (CAD12)	MCB DFILE 12
MCB data file 12	X'180' of CAT (CAD13)	MCB DFILE 13
MCB data file 13	X'184' of CAT (CAD14)	MCB DFILE 14
MCB data file 14	X'188' of CAT (CAD15)	MCB DFILE 15
MCB data file 15	X'18C' of CAT (CAD16)	MCB DFILE 16
MCB data file 16	X'190' of CAT (CAD17)	MCB DFILE 17
MCB data file 17	X'194' of CAT (CAD18)	MCB DFILE 18
MCB data file 18	X'198' of CAT (CAD19)	MCB DFILE 19
MCB data file 19	X'19C' of CAT (CAD20)	MCB DFILE 20
MCB data file 20	X'1A0' of CAT (CAD21)	MCB DFILE 21
MCB data file 21	X'1A4' of CAT (CAD22)	MCB DFILE 22
MCB data file 22	X'1A8' of CAT (CAD23)	MCB DFILE 23
MCB data file 23	X'1AC' of CAT (CAD24)	MCB DFILE 24
MCB data file 24	X'1B0' of CAT (CAD25)	MCB DFILE 25
MCB data file 25	X'1B4' of CAT (CAD26)	MCB DFILE 26
MCB data file 26	X'1B8' of CAT (CAD27)	MCB DFILE 27
MCB data file 27	X'1BC' of CAT (CAD28)	MCB DFILE 28
MCB data file 28	X'1C0' of CAT (CAD29)	MCB DFILE 29
MCB data file 29	X'1C4' of CAT (CAD30)	MCB DFILE 30

Figure 152 (Page 2 of 3). Locating and Identifying Control Blocks, Tables, and Areas in the Fixable Area

Abbreviated * Mnemonic of Table or Area	Pointer to or Address of the Table/Area	Identifier in Translated Dump
MCB data file 30	X'1C8' of CAT (CAD31)	MCB DFILE 31
MCB data file 31	X'1CC' of CAT (CAD32)	MCB DFILE 32
MCB data file 32	X'1D0' of CAT (CAD33)	MCB DFILE 33
Remote message contr. blk.	X'11C' of CAT (CARM)	MSCB
Master external device control block (MEDCB)	X'138' of CAT (CAEDCB)	MEDCB
Communicator info block	X'134' of CAT (CACI)	CIB
Communicator info block 2	X'144' of CAT (CACI2)	CI2
Physical work space (PWS)	R8 in a TCB for a task in a physical routine	
Physical data area	X'00' of a PWS X'04' of a PWS	Virtual address Real address
Relative number of current queue record	X'18C' - X'18F' of TCB (TCQW)	
Queue record area	X'190'-X'193' of a TCB (TCQV)	Virtual address
Queue record identifier	X'2A' of a queue record (QRQI)	R,L,P,F,D,B,I
Size of phys. data buffer	X'514' of CAT (CABLBF)	
Size of log. data buffer	X'518' of CAT (CABLDB)	
Relative queue record number of master record	X'20' - X'23' of DMB (QCMW)	
Relative queue record no. of next free queue record	X'30C' - X'30F' of DMB (MRQFRNO)	
SLI work space (SLWA)	X'40' of PART.CONTR.BLOCK (PDSL)	
SNA control block	X'120' of CAT (CASM)	SNCB
PNET master control block	X'130' of CAT (CAPN)	PNCB
Asyn service anchor block	X'128' of CAT (CAAB)	ASWS
Virt. storage control blck	X'140' of CAT (CAVS)	VSCB
Dyn. part. control clock	X'13C' of CAT (CADPCB)	DPCB
Trace information block	X'12C' of CAT (CATK)	TIB
Curr trace area descriptor	X'14' of TIB (TIBPTRC)	
Free trace area entry	X'50' of TIB (TIBCFTE)	
First node control block	X'10' of PNCB (PNCBNCB)	NCB
TCB of PNET Driver	X'14' of PNCB (PNCBTLD)	
VTAM Driver control block	X'20' of PNCB (PNCBVDCB)	VDCB
PNET TCP Driver control block	X'28' of PNCB (PNCBTDCB)	TDCB
PNET SSL Driver control block	X'2C' of PNCB (PNCBSDCB)	SDCB
Start of fixable area	X'48' of CAT (PAFA)	
Start of pageable area	X'4C' of CAT (PAVA)	
Start of task select. list	X'14' of WCB (TMTN of IPW$$NU)	
Tape control block	X'199' of TCB. (TCTA)	TBB
TCB of last attached auxiliary command proc.	X'24C' of CAT (CAOP)	If none exist, addr. of perm. command proc.
TCB of last attached execution processor task	X'260' of CAT (CAEX)	
TCB of highest priority task in task selection list	X'14' - X'17' of WCB (TMTN of IPW$$NU)	
TCB of RJE,BSC line or SNA manager or PNET driver	X'23C' of CAT (CALM)	If not present then WCB addr.
TCB for most recently attached writer task	X'25C' of CAT (CARW)	
TCB for most recently attached RJE task	X'254' of CAT (CARJ)	
TCB for most recently attached reader task	X'264' of CAT (CARR)	
Task register save area	X'40' - X'77' in any TCB (TCTR)	

Figure 152 (Page 3 of 3). Locating and Identifying Control Blocks, Tables, and Areas in the Fixable Area

Abbreviated * Mnemonic of Table or Area	Pointer to or Address of the Table/Area	Identifier in Translated Dump
Task selection list	X'14' - X'17' of WCB; then X'14' - X'17' of each TCB (TCTN) until return to WCB	Recognize each TCB by its descriptor.

* Refer to "List of Abbreviations" on page 799.

In the VSE/AF GETVIS Area: Control blocks, tables, and areas in the VSE/AF GETVIS area can be found by reference to Figure 153.

Figure 153. Locating and Identifying Control Blocks, Tables and Areas in the VSE/AF GETVIS Area

Abbreviated * Mnemonic of Table or Area	**Pointer to or Address of the Table/Area**	**Identification in the Translated Dump Output**
COCB	X'40' - X'43' of SNCB (SNCA)	COCB
LRCB	X'38' - X'3B' of SNCB (SNLR)	LRCB
LUCB	X'29' - X'2B' of SUCB (SUL1L)	LUCB
	X'39' - X'3B' of SUCB (SUL2L)	
	X'49' - X'4B' of SUCB (SUL3L)	
	X'59' - X'5B' of SUCB (SUP1L)	Only if appropriate device is processing
	X'69' - X'6B' of SUCB (SUR1L)	
	X'79' - X'7B' of SUCB (SUX1L)	
	X'89' - X'8B' of SUCB (SUC1L)	
SUCB	X'14' - X'17' of SNCB (SNFS)	
RMCB	X'20' - X'23' of SNCB (SNRM)	
CI Put-Account	X'78' - X'7B' of ACB (AFWAF)	CI FBA P/A
CI Save-Account	X'7C' - X'7F' of ACB (AFWASA)	CI FBA S/A
Network definition table	X'18' of PNCB (PNCBNDT)	NDT
Temporary NAT table	X'24' of PNCB (PNCBTNT)	
External device cntl block (1st in chain)	X'10' of MEDCB (MEDCBFEL)	EDCB
Communicator info element (1st in chain)	X'5C' of CIB (CIBFCIE)	CIE

* Refer to "List of Abbreviations" on page 799.

Identifying the Start of a CSECT

Each control section within the VSE/POWER code is identified by a 16-byte control section descriptor in the following format.

- The alphameric name assigned to the control section
- The level identifier for this release or modification level
- The date of the last compilation for this phase or last applied APAR number.

Establishing the Level of a CSECT

The level of a routine (Physical, Logical, Execution, Function, Service) can be established by the first two/three characters of its CSECT name identified in a dump. For example, if the contents of register 12 in a TCB points to an address within CSECT name AQCS, the calling routine (AQCS) is at FUNCTION level.

Determining the Active Routine and Analyzing Register Save Areas

It is important to know the routine in which a task is executing in order to be able to analyze the meaning of the contents of the registers saved.

The contents of R12 in the TRSA in a TCB belonging to a task that is not in R state will address the instruction that will be next executed when the task is given control. The routine or CSECT in which this instruction is located can be identified in a dump by means of the storage descriptor.

Figure 147 on page 715 to Figure 150 on page 718 show the relationship between the LRSA in TCB and the DLRSA or second LRSA, which depends on the calling sequence of VSE/POWER routines.

Analyzing Event Control Blocks (ECBs)

Several control blocks are equipped with ECBs, the condition of which may be important to problem analysis. The possible conditions are:

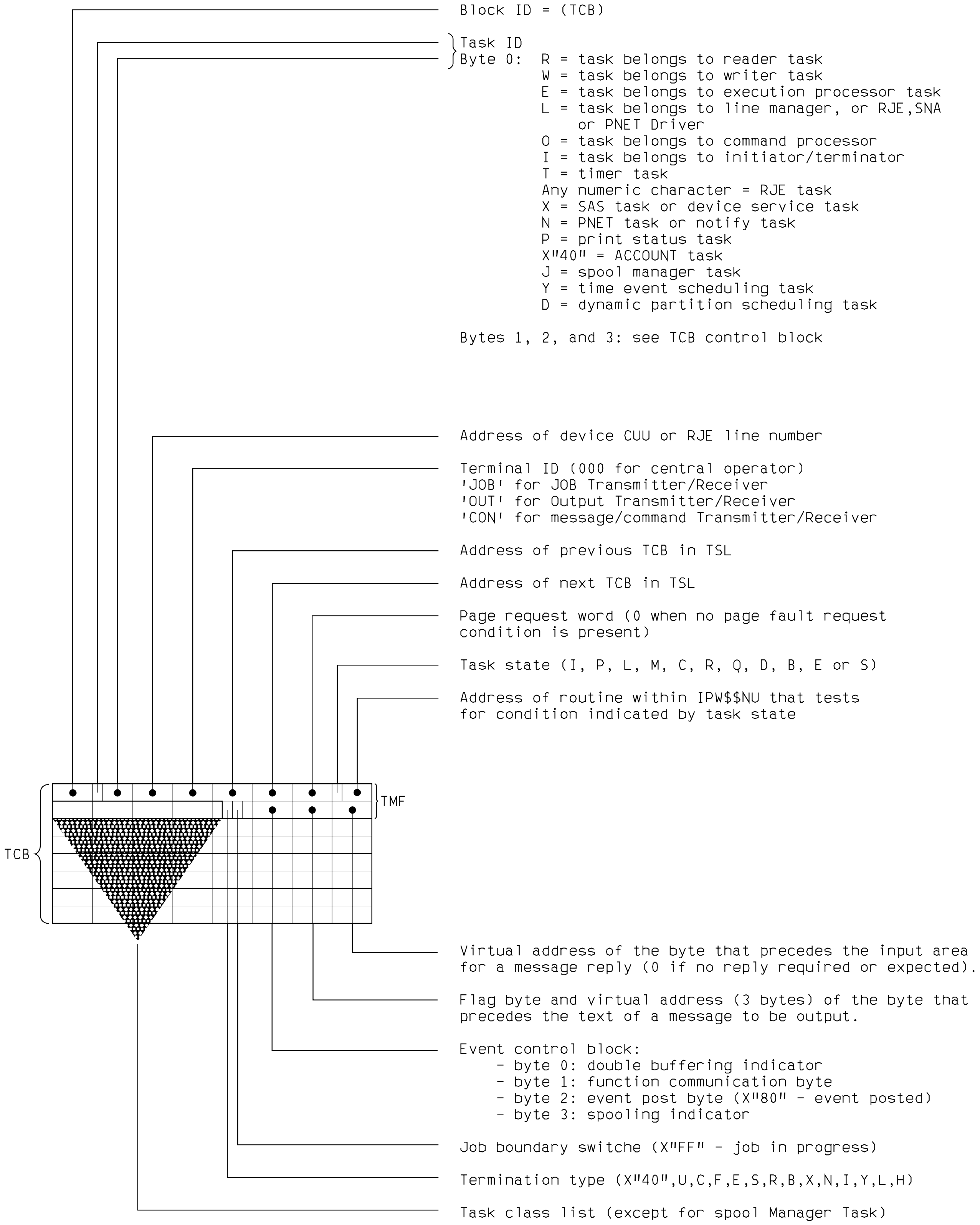
- Posted - bit 16 on (1)
- Unposted - bit 16 off (0). See Appendix B, “Summary of ECB Usage (4 and 8-Byte).”

Using Buffer Control Words

The four bytes immediately in front of any area contain the address of the task control block of the task which reserved the area.

Analyzing TCBS

Figure 154 on page 736 and Figure 155 on page 737 are for quick reference only.



Block ID = (TCB)

Task ID
Byte 0: R = task belongs to reader task
W = task belongs to writer task
E = task belongs to execution processor task
L = task belongs to line manager, or RJE,SNA or PNET Driver
O = task belongs to command processor
I = task belongs to initiator/terminator
T = timer task
Any numeric character = RJE task
X = SAS task or device service task
N = PNET task or notify task
P = print status task
X"40" = ACCOUNT task
J = spool manager task
Y = time event scheduling task
D = dynamic partition scheduling task

Bytes 1, 2, and 3: see TCB control block

Address of device CUU or RJE line number

Terminal ID (000 for central operator)
'JOB' for JOB Transmitter/Receiver
'OUT' for Output Transmitter/Receiver
'CON' for message/command Transmitter/Receiver

Address of previous TCB in TSL

Address of next TCB in TSL

Page request word (0 when no page fault request condition is present)

Task state (I, P, L, M, C, R, Q, D, B, E or S)

Address of routine within IPW$$NU that tests for condition indicated by task state

Virtual address of the byte that precedes the input area for a message reply (0 if no reply required or expected).

Flag byte and virtual address (3 bytes) of the byte that precedes the text of a message to be output.

Event control block:
- byte 0: double buffering indicator
- byte 1: function communication byte
- byte 2: event post byte (X"80" - event posted)
- byte 3: spooling indicator

Job boundary switche (X"FF" - job in progress)

Termination type (X"40",U,C,F,E,S,R,B,X,N,I,Y,L,H)

Task class list (except for spool Manager Task)

Figure 154. General Meaning of the Task Management Fields

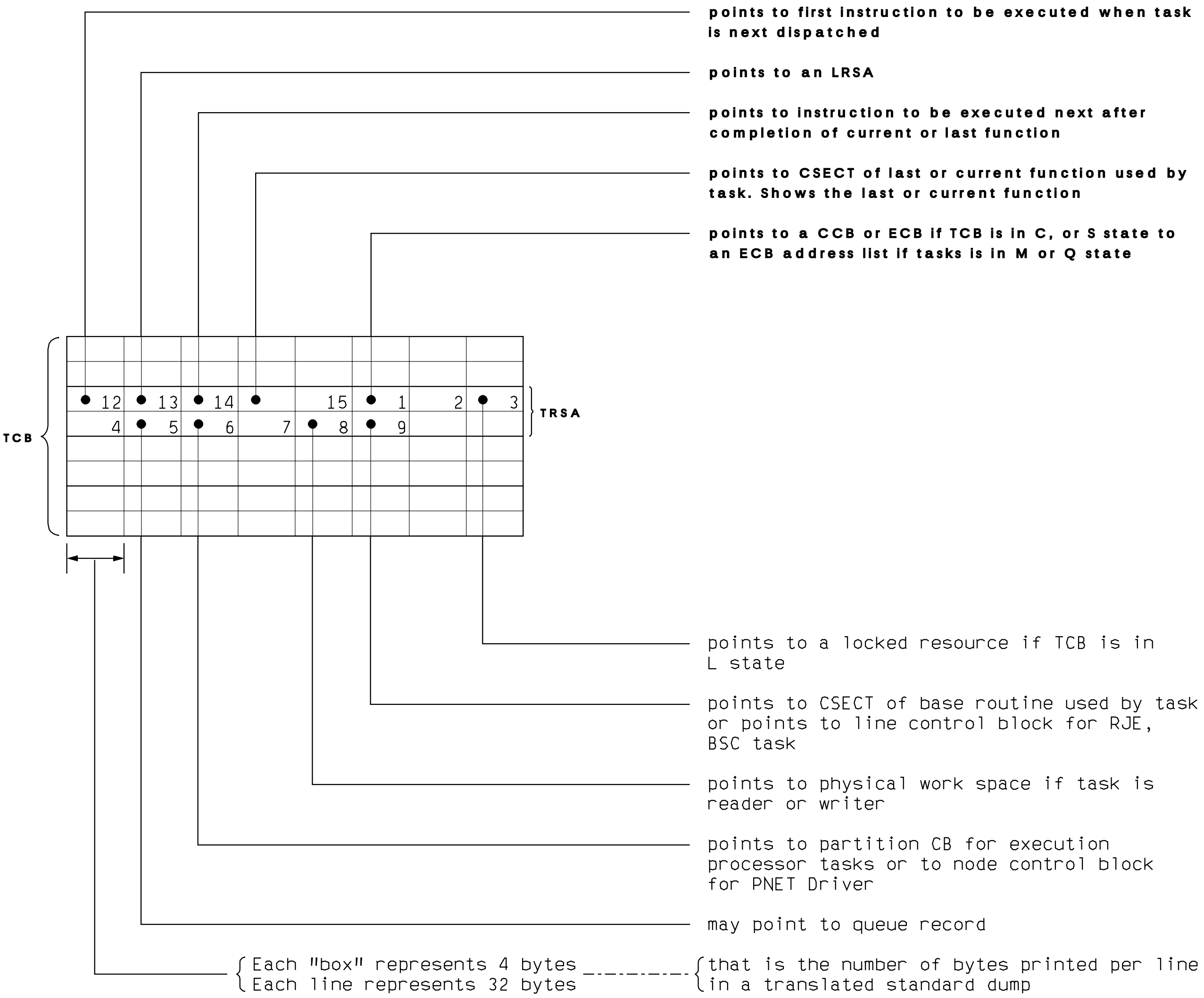

points to first instruction to be executed when task is next dispatched
points to an LRSA
points to instruction to be executed next after completion of current or last function
points to CSECT of last or current function used by task. Shows the last or current function
points to a CCB or ECB if TCB is in C, or S state to an ECB address list if tasks is in M or Q state
12
13
14
15
1
2
3
4
5
6
7
8
9
TRSA
TCB
points to a locked resource if TCB is in L state
points to CSECT of base routine used by task or points to line control block for RJE, BSC task
points to physical work space if task is reader or writer
points to partition CB for execution processor tasks or to node control block for PNET Driver
may point to queue record
Each "box" represents 4 bytes
Each line represents 32 bytes
that is the number of bytes printed per line in a translated standard dump

Figure 155. General Meaning of Fields in the TRSA

RJE,BSC and PNET Telecommunication Trace Facility

The following provides a combined I/O and buffer content trace description (and is duplicated in the *VSE/POWER Networking*, SC33-6735 publication).

The trace is a useful tool which aids the user in problem determination. It also is a debugging aid that enables the system programmer or an IBM program system representative to locate the cause of an failure.
With the operator's console log and a dump, the output from the trace area provides enough information to locate internal PNET or RJE,BSC problems more easily and permits the re-construction of I/O sequences.

A trace record is written in following events:

RJE,BSC I/O completed
PNET BSC/CTC I/O completed
PNET SNA SEND or RECEIVE request completed
PNET SNA RECEIVE request completed
PNET TCP (any) socketcall started, tested or completed

The trace records generated by VSE/POWER are recorded in wrap-around fashion in main storage. The amount of storage allocated for the trace table is specified in the TRACESZ=xxx parameter of the POWER macro at generation time. The value specified for TRACESZ should ensure that information is not destroyed because to the wrapping in the trace table. The value should also reflect the amount of storage available.

The operator activates the trace recording for a RJE,BSC or PNET line by specifying the TRACE operand in the PSTART command. The trace recording keeps active until the RJE,BSC line or the connection to the other node is stopped.

Each trace record is 256 bytes long. The following is a list of possible types of trace records:

- RJE,BSC trace record
  The format of the trace record is described in “RJE,BSC Trace Record”
- PNET BSC/CTC trace record
  The format of the trace record is described in “PNET BSC/CTC Trace Record”
- PNET SNA SEND/RECEIVE trace record
  The format of the trace record is described in “PNET SNA Trace Record”
- PNET TCP trace record
  The format of the trace record is described in “PNET TCP Trace Record”
- PNET SSL trace record
  The format of the trace record is described in “PNET SSL Trace Record”

The trace records can be examined by displaying or taking a dump of the main storage location containing the trace area.

Optionally whenever the trace area is full it is written onto the VSE/AF DUMP library. The operator is informed when a trace area is successfully written into the user defined DUMP library. The various dump members can then be printed out by the appropriate VSE/AF utility. Trace logging is requested by means of UPSI 001 at VSE/POWER start-up time or can be dynamically requested while VSE/POWER is running by entering the PSTART DUMPTR commend.

The trace area is divided into two parts, referred as primary and alternate trace area. Both trace areas have the same size (integer number of page). When the primary trace area is full, VSE/POWER automatically swaps to the alternate trace area and starts filling that up. If now the alternate trace area is filled up, the primary trace area is addressed again and used for recording.

The trace area can be found by first locating the trace information block (TIB). The trace information block contains among others a pointer to a two-word trace area descriptor of the trace area currently in use at displacement X'14'. The first word of the trace area descriptor contains the trace area begin address while the second word contains the end address. The pointer to the next free trace area entry is stored at displacement X'50' of the TIB. The PDISPLAY TRINFO command can be used to get the start and end address of the entire trace area as well as the current free trace area address. See Figure 54 on page 143 for the control block structure.

RJE,BSC Trace Record

(refer to the Appencix D of the *VSE/POWER Remote Job Entry*, SC33-6734 publication)
(The DSECT 'TRACEDS' is found in the module IPW$$LM)

PNET BSC/CTC Trace Record

(refer to the Appencix D of the *VSE/POWER Networking*, SC33-6735 publication)
(The DSECT 'TRACENT' is found in the module IPW$$LD1)

PNET SNA Trace Record

(refer to the Appencix D of the *VSE/POWER Networking*, SC33-6735 publication)
(The DSECT 'TRACENT' is found in the module IPW$$LD2)

PNET TCP Trace Record

(refer to the Appencix D of the *VSE/POWER Networking*, SC33-6735 publication)
(The DSECT 'TRCENTRY' is found in the module IPW$$TD)

PNET SSL Trace Record

(refer to the Appencix D of the *VSE/POWER Networking*, SC33-6735 publication)
(The DSECT 'TRCENTRY' is found in the module IPW$$SD)

PNET BSC/CTC/TCP I/O Logging on Console

(refer to the Appencix D of the *VSE/POWER Networking*, SC33-6735 publication)

Hardware Error Recording

Error recording is part of ERP processing. The error data is placed in the system error recorder file (SYSREC) for subsequent editing and printing. Error recording takes place under following conditions:

- BSC RJE Line
  - *Permanent errors* -- errors that are either unrecoverable or errors from which VSE/POWER error recovery processing failed to recover.
  - *Counter overflow* -- written whenever the SIO (START I/O) counter, the temporary error counter, or one of the device statistics table counters is about to overflow.
  - *End-of-day* -- written whenever a BSC line is stopped.
- BSC/CTC PNET Line

  A unit check record is written onto SYSREC whenever:
  - A channel program, channel protection, channel data, channel control, channel interface, or channel chaining check occurs,
  - A command reject or CCW chain does not end on a read CCW,
  - A recoverable line error (other than time out) occurs,
  - A remote or local node detects a sequence error.

VSE/POWER Disk Dump Program

This program enables any of the VSE/POWER files (account, queue, data) to be dumped on a line printer or tape assigned to SYSLST. An option is also provided to dump a specific queue record with its associated DBLK groups. In a shared-spooling environment the queue-control area could also be dumped.

For more information and how to execute the VSE/POWER disk dump program refer to *VSE/POWER Administration and Operation*.

Establishing the Last Command Issued

The last command issued by the central operator can be seen printed in the translated part of a dump within the permanent command processor task control block, recognized by the storage descriptor CPB.

An Aid to Eliminate Functions

It may be useful to have several different generation tables cataloged in the library with at least a default version as originally supplied to the user. The various versions act as a debugging aid to eliminate the various optional functions. (e.g. Source Library Inclusion is only present if SUBLIB= or MEMTYPE= was specified in the POWER generation).

Problems Related to VTAM

If a problem occurs where VTAM is involved, please consult the appropriate VTAM diagnostic manuals.

System Dump Containing the VSE/POWER Partition

For a full description of a system dump, refer to *VSE/ESA Diagnosis Tools*, SC33-6614. See Figure 140 on page 435 for a pictorial representation of the VSE/POWER partition.

Appendix A. VSE/POWER Status Bytes in the VSE/AF Supervisor

SYSCOM*

Location X'42' (IJBFLG03) contains a flag byte:

```
X'04' (IJBPOWA) = VSE/POWER initialized
```

Location X'5C' - X'5F' (IJBPWR) contains the address to the VSE/POWER control address table, which is included in the VSE/POWER nucleus (IPW$$NU) if VSE/POWER is initiated.

Location X'154' - X'157' (IJBPSYSI) contains the address to the VSE/POWER information (SYSID and PNET node name). Set up during VSE/POWER initialization.

Location X'163' (IJBFLG09) contains a flag byte:

```
X'01' (IJBPOWMP)= VSE/POWER Multiprocessor Support enabled (SET WORKUNIT =PA)
```

Partition COMREGS*

Location X'A0' - X'A3' (POWPCB) contains the address of the VSE/POWER partition control block (0 if no partition control block exists for this partition).

Location X'A4' (POWFLG1) contains VSE/POWER flags:

```
X'80' (POWACT) = VSE/POWER accounting support
X'40' (POWUPART) = This partition under control of VSE/POWER
X'20' (POWPART) = This partition is the VSE/POWER partition
X'10' (POWPDORM) = Partition is dormant
X'08' (POWWPART) = Partition waiting for work state
X'04' (POWBAM) = Used for BAM TRC interface to VSE/POWER
X'02' (POWHIGH) = VSE/POWER has lower priority
X'01' (POWSPERR) = Write error on VSE/POWER data file
```

Location X'A5' (POWFLG2) contains VSE/POWER flags:

```
X'80' (POWUNBCH) = VSE/POWER automatic unbatch indication
X'40' (POWUNBTS) = Feedback for TSTOP
X'20' (POWINTER) = VSE/POWER in termination
X'10' (POWINTFL) = VSE/POWER internal flush
X'08' (POWPFRC) = VSE/POWER PEND FORCE
X'04' (POWIGLOG) = VSE/POWER LOG=NO option
X'02' (POWWRONL) = VSE/POWER Writer-only partition
X'01' (POWDREC) = VSE/POWER Dummy Record
```

Partition Control Block Extension (PCE)*

Almost the complete PCE (about 36 bytes) is used as an interface between VSE/AF and VSE/POWER.

* Refer to *VSE/ESA Diagnosis Tools*, for a full description and locations of the above VSE/AF Supervisor control areas.

Appendix B. Summary of ECB Usage (4 and 8-Byte)

4-Byte ECB usage: is summarized in Figure 156

Figure 156. Summary of ECB Usage

ECB in:	Posted by: (Phase)	Unposted by:	Use when posted:
ACB	IPW$$GA/IPW$$SF	IPW$$PA/PF	Account file is empty
CAT	Appendage	Task select.	Indicates work-to-do for VSE/POWER.
SCB	IPW$RLW	IPW$RSW	Work space is avail.
VSCB	IPW$RLV	IPW$RSV	Virtual storage avail.
DMB	IPW$$FQ	IPW$$RQ, IPW$$PD	Queue space is avail.
TCB (CP)	IPW$$I7	IPW$$CM	Indicates that IPW$$I7 has sent information to IPW$$CM.
TCB (LD)	all PNET tasks IPW$$AQ, IPW$$MS, IPW$$CPS, IPW$$CP	IPW$$LD	If work is to do for PNET driver.
TCB (LMGR)	• Channel End Appendage • line start • line stop	IPW$$LM	Work-to-do for line manager.
TCB (OB)	IPW$$IB, IPW$$MP, IPW$$SN	IPW$$OB	Indicates that trans. to SNA terminal which was previously suspended is to continue.
TCB (SN)	VTAM at completion of a RECEIVE ANY	IPW$$SN	Indicates that IPW$$SN must attach IPW$$IB.
TCB (SN)	IPW$$SN, IPW$$LN, IPW$$IB, IPW$$OB, IPW$$MP, IPW$$LF, IPW$$VE, IPW$$LH, IPW$$MS	IPW$$SN	Indicates work-to-do for IPW$$SN.
SRB	IPW$$AS (Subtask)		Indicates that service request is processed.
ASAB	IPW$$AS	IPW$$AS (Subtask)	Indicates that service request is waiting to be processed.
DPCB	IPW$$CS	IPW$$DP	Indicates that PSTART command for dynamic partition is processed.

8-Byte ECB usage: an entry in the master class table area can be used as an 8-byte ECB. In that case the address of the entry is contained in the task class list (ECB list) in the TCB. When the ECBs in the RDR, LST, or PUN class are posted (by IPW$$AQ), they indicate that an active entry exists in the class chain represented by this class table entry. These ECBs are unposted by IPW$$NQ.

For example, assume a TCB for a LST task in the queue state as shown in Figure 157.

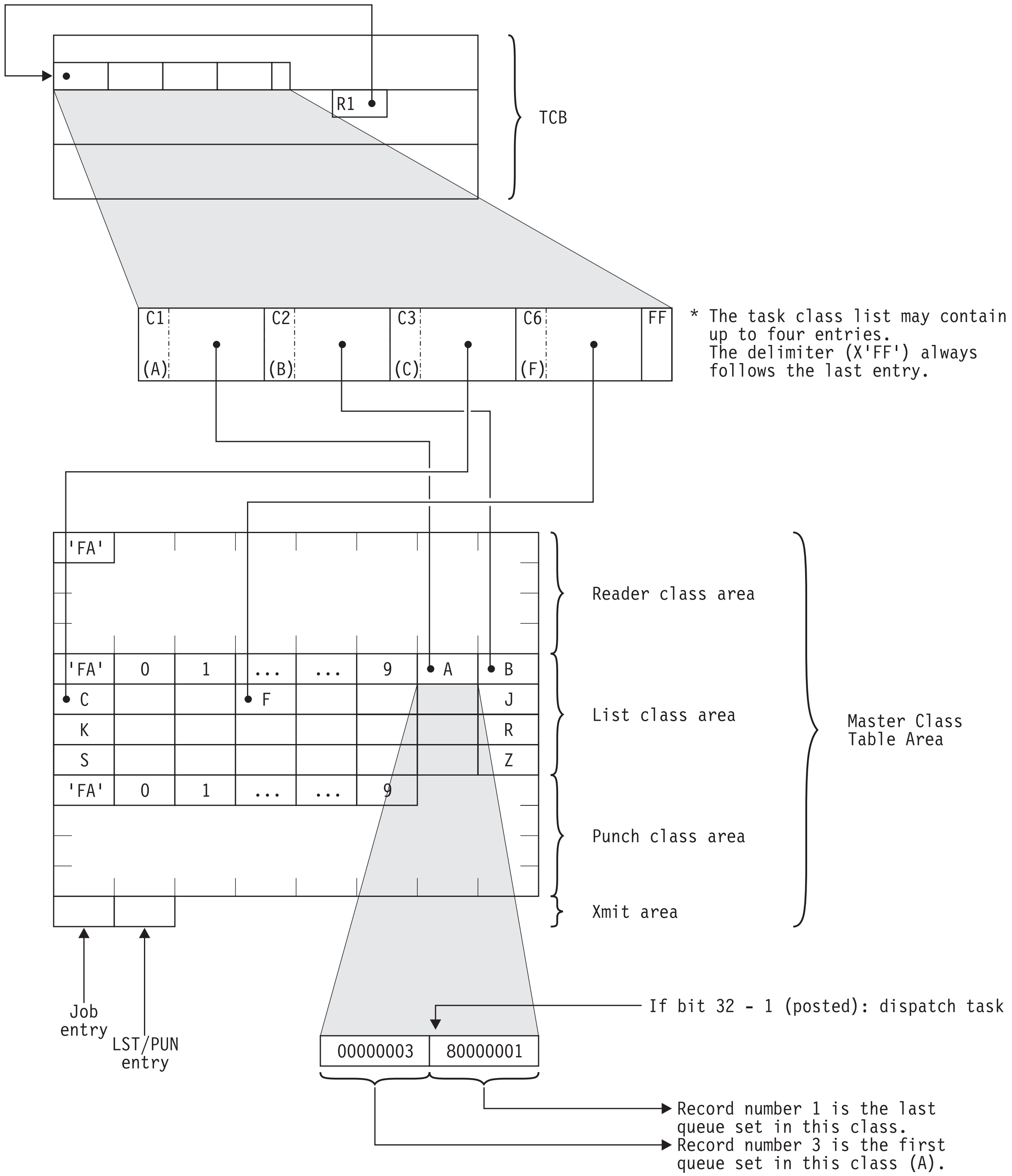

R1
TCB
C1
(A)
C2
(B)
C3
(C)
C6
(F)
FF
* The task class list may contain up to four entries. The delimiter (X'FF') always follows the last entry.
'FA'
Reader class area
'FA'
0
1
...
...
9
A
B
C
F
J
K
R
S
Z
List class area
Master Class Table Area
'FA'
0
1
...
...
9
Punch class area
Xmit area
Job entry
LST/PUN entry
If bit 32 - 1 (posted): dispatch task
00000003
80000001
Record number 1 is the last queue set in this class.
Record number 3 is the first queue set in this class (A).

Figure 157. Relationship Between Classes in the TCB and the Master Class Table in the DMB

Appendix C. VSE/POWER Internal Macros

VSE/POWER provides a comprehensive set of services which aid the VSE/POWER tasks in performing their respective tasks in a efficient manner without burdening the programmer with needless details. These services are requested by the VSE/POWER tasks through the use of macros and should not be used in code outside the control of the VSE/POWER dispatcher.

Some of the services are provided by inline code expansion wherever the macro instruction is used. The remaining services are provided by routines which are either part of the VSE/POWER nucleus (IPW$$NU) or other modules. These routines are linked to by code generated wherever the macro instruction is used. At execution time the macro expansion passes information to the routine to specify the exact nature of the service to be performed. This information is broken down into parameters and, in general, passed to the routine through general purpose registers called parameter registers.

The VSE/POWER macros are presented in this section by means of macro instruction description, each of which contains an illustration of the macro format. Parameters are specified by operands in the macro instruction. Operands are of two types, positional operands and keyword operands. Keyword operands can be written in any order, but they must be written to the right of any positional operand in the macro instruction.

Coding Aids

The symbols [ ], { }, and _ are used in this publication to help define the macro instructions. These symbols are not coded; they are only to indicate how a macro instruction is to be written; their general definitions are given below:

[] indicates optional operands. The operand enclosed in the brackets may or may not be coded, depending on whether or not the associated option is desired.

{ } indicates that a choice must be made. One of the operands within the braces must be coded, depending on which of the associated services/functions is desired.

_ an underlined parameter indicates the default if the parameter is not coded.

Macro Notation

The VSE/POWER macros are written in the assembler language and, as such, are subject to the rules contained in OS/VS - DOS/VS - VM/370 Assembler language.

The following describes the meaning of each notation used:

(reg) When this notation is shown, a general register must be coded. It is assumed that designated register contains the address or value.

(Rx) When this notation is used, it is assumed that the general register x already contains the address or value, unless otherwise specified in the description of the macro.

Format of Internal Macros

IPW$ALN - Align to Storage Boundary

The IPW$ALN macro causes to align the storage to the specified boundary and to fill the storage with X'FF'. In most cases, the macro is used to generate a patch area at the end of the module.

```
[name]    IPW$ALN  boundary
```

boundary specifies the requested storage boundary as one of the following:

LINE causes to align to next x'20' boundary

PARA causes to align to next x'100' boundary

PAGE causes to align to next x'800' boundary

address causes to align to the specified storage address. The address can be specified in hex notation or as decimal value.

IPW$AJ# - Assign New VSE/POWER Job Number

The IPW$AJ# macro assigns a new VSE/POWER job number. The macro expansion generates inline code.

Registers 0 - 3 are destroyed by execution of the macro.

```
 [name]   IPW$AJ#   address|(reg)[,LOCK=YES|NO]
```

address specifies the address of a 2-byte field to hold the job number assigned by VSE/POWER. If register notation is used the designated register must point to a 2-byte field.

LOCK YES causes the macro expansion to lock the DMB for exclusive use before updating the VSE/POWER job number.

NO means that the issuing routine already owns the DMB and that therefore no lock of the DMB is necessary.

IPW$AQS - Add Queue Entry to Class Chain

The IPW$AQS macro is used to add a queue entry, pointed to by the TCB, to the appropriate class chain according to its priority. Depending on the queue record id, contained in the queue record area, the queue entry is added to the RDR, LST or PUN queue. If the queue entry is destined for a remote node in the network, the queue entry is automatically added to the XMT queue.

Any VSE/POWER task waiting for work is posted when the added queue entry meets its processing criteria.

Registers 14 and 15 are used as linkage registers.

```
[name]   IPW$AQS   [KEEP][,LOCK=YES|NO]
```

KEEP specifies that the queue entry is already queued in one of the class chains. It causes to write back the queue record of the queue entry addressed by the TCB.

LOCK YES specifies that the add queue entry routine performs a DMB release operation after processing the requested function; this is the default.

NO means that the calling task already owns the DMB and that the DMB should not be released.

IPW$ATT - Attach VSE/POWER Task

The IPW$ATT macro is used to attach a new VSE/POWER task. Register 1 must address the TCB of the task to be attached; the service routine assumes that the TCB is properly built.

Registers 0 - 3 are destroyed by execution of the macro instruction.

```
[name]    IPW$ATT  [R1][,symbol]
```

symbol specifies the name of the VSE/POWER phase to which control is given when the task begins its execution. This is the name of the phase as defined in the CAT with the first two characters stripped off. If the parameter is omitted, register 3 must have previously loaded with the address of the phase.

IPW$BUF - Invoke PNET Buffer Service

The IPW$BUF macro is used to obtain, queue, release or purge a TP buffer used by PNET. The macro expands into a linkage to the buffer service routine (IPW$$BS).

A set of buffer service routines are provided to queue any incoming TP-buffer in the 'received queue', to queue any output buffer in the 'to-be-sent queue', and to supply buffers for both transmitters and receivers.

The buffers required to process BSC nodes are provided from real storage, while those required to process SNA nodes are provided from virtual storage (GETVIS).

Registers 14 and 15 are used as linkage registers; registers 0 and 1 are destroyed by execution of the macro instruction.

```
[name]    IPW$BUF  MODE=IN|OUT[,REG=(reg)][,WAIT=YES|NO]
                   ,TYPE=GET|RELEASE|FREE|QUEUE|CNTRL|PURNT|PURNR|MSG
```

REG specifies the address of the buffer that is to be referenced. If not specified then the address from the NCB will be used. This keyword is only valid for TYPE=QUEUE,MODE=OUT.

WAIT specifies whether the task wants to wait if there is either insufficient storage to satisfy the request or no TP buffer in the 'received input queue'.

YES specifies that the task wants to be placed in the wait state until a buffer is available. YES is the default if the parameter is omitted.

NO means that control will be returned directly to the calling task, with or without a buffer being available.

MODE=IN specifies that the buffer to be processed is a input buffer.

FREE causes the TP buffer addressed by register 1 to be queued as first entry in the 'free input chain'. The calling task's FCS bit is set in order to resume a data stream which has been suspended, prior to receiving the maximum number of buffers.

Note: This function is invoked by the PNET driver and the receiver.

GET causes the buffer addressed by the head pointer of the 'received queue' to be dequeued and its address to be returned in register 1. If the 'received queue' is empty and WAIT=NO is specified, then a return is made to the user with register 1 containing zero to indicate that no buffer was available. If the 'received queue' was empty and WAIT=YES is specified, then the task is put into a wait for a BSC/PNET event.

Note: This function is only invoked by the receiver.

QUEUE causes the input buffer being received, addressed by register 1, to be queued as last entry in the 'received input queue' for the task, checks the number of buffers in the queue against the maximum value specified in the MAXBUF parameter for the node, and if the maximum is reached sets a suspend for this task.

Note: This function is invoked only by the PNET driver when an input buffer is received successfully.

RELEASE causes all buffers up to the maximum specified by MAXBUF, to be freed from the 'free input queue' and to be returned to the VSE/POWER storage pool. If the number of buffers in the 'free input queue' is less than MAXBUF then all are freed except for one.

Note: This function is only invoked by the PNET driver.

MODE=OUT specifies that the buffer to be processed is an output buffer.

CNTRL causes a small TP buffer used for a NJE control record to be reserved and its address to be placed in register 1, upon return. The buffer is marked to be freed after successful transmission. If WAIT=NO is specified or defaulted and insufficient storage is available to fulfill the requirement then return is made to the user with register 1 as zero. If WAIT=YES is specified, the task is put into wait until space becomes available.

FREE causes the output buffer currently being sent, addressed by register 1, to be queued as first entry in the 'free output queue' of the related NCB task entry.

If one of the following conditions arises then the buffer is released and its storage returned to the VSE/POWER storage pool.

- If the 'release-after-sent' flag is set in the buffer header.
- If the buffer owner is the PNET driver.
- SIGNOFF is in process or a line error has occurred.

If the 'short-on-buffer' condition is set in the NCB task entry (NCBEST2 = NCBESOB) then the task must also be posted.

Note: This function is only invoked by the PNET driver after a buffer has been successfully sent.

GET causes a TP-buffer to be allocated from the 'free output queue' and its address to be returned in register 1. If no buffer is in the 'free output chain' and the maximum number of acquired output buffers is not exceeded for the calling task, a new TP-buffer is acquired and the number of buffers in use is increased.

If no buffer is available in the 'free output chain' and the number of output buffers for this task has not reached the maximum value, an attempt to acquire another buffer from the VSE/POWER storage pool is made.

If no storage is available and this is not the first buffer, then a wait is made for a buffer to be freed by the FREE option of the IPW$BUF macro.

If this was the first buffer to be acquired, then the reserve work space is made with the WAIT option.

If the maximum number of buffers have already been acquired then the task is put into wait for a buffer to be freed by the FREE option of the IPW$BUF macro.

MSG is the same as CNTRL but provides larger buffers to be used for messages.

PURNR causes buffers queued for transmission, by the receiver, to be removed from the 'to-be-sent-queue' dependent on the following conditions:

- Register 1 is zero, then ALL buffers are removed from the queue.
- Register 1 is non-zero, then only buffers belonging to this task are removed from the queue.

If the buffer has the 'release-after-sent' flag set then the buffer is returned to the VSE/POWER storage pool. All other buffers are queued into the 'free output queue'.

PURNT causes buffers queued for transmission, by the transmitter, to be removed from the 'to-be-sent-queue' dependent on the following conditions:

- Register 1 is zero, then ALL buffers are removed from the queue.
- Register 1 is non-zero, then only buffers belonging to this task are removed from the queue.

If the buffer has the 'release-after-sent' flag set then the buffer is returned to the VSE/POWER storage pool. All other buffers are queued into the 'free output queue'.

QUEUE causes the TP-output buffer, addressed by register 1, to be queued as the last entry in the 'to-be-sent chain'. If the buffer just queued is the first in the queue, then the PNET driver is posted.

If a line error has been detected or a 'SIGN-OFF' record has been received, then the buffer is added to the 'free output queue'.

RELEASE causes all TP-buffers in the 'free output chain' (anchored to this NCB task entry), to be released and returned to the VSE/POWER storage pool.

IPW$CAF - Close Account File

The IPW$CAF macro is used to close the account file and prepare the account file for write operation.

Note: The account file must have been previously opened via the IPW$OAF macro instruction otherwise the action of the IPW$CAF macro instruction is unpredictable.

Registers 14 and 15 are used as linkage registers.

```
[name]    IPW$CAF  {CLOSE|ERASE|KEEP}
```

CLOSE specifies to close the account file and to prepare the account file again for write operation.

ERASE specifies to erase the account file. An EOF record is written as first record on each track (CKD), or when the account file resides on a FBA device an EOF CI is written as first CI of the account file.

KEEP specifies to return the account file for write operation.

IPW$CLI - Close Logical Interface

The IPW$CLI macro is used to close the interface to the logical routine (IPW$$LO, IPW$$LR, IPW$$LW). It returns the additional save area, obtained by means of the IPW$OLI macro when the interface was opened, to the VSE/POWER storage pool.

Registers 0 - 3 are destroyed by execution of the macro.

```
[name]    IPW$CLI  [LO]
```

LO specifies that the interface to the logical output spooler (IPW$$LO) should be closed and the save area appendix storage, if present, released.

Note: NO operand should be specified when 'closing' the interface to the logical reader or writer.

IPW$CPY - Provide Copyright

This macro generates an object code readable copyright constant by the following inclusions:

1. a branch instruction to bypass the subsequent copyright constant
2. a 44 byte copyright constant of the following layout:

   ```
   '5686-066 (C) COPYRIGHT IBM CORP. 19xx, 19yy '
   ```

   where '19xx' is the year of first availability
   and '19yy' is the year of availability since the code was changed last.
3. OR a 44 byte copyright constant of the following layout:

   ```
   '5686-066 (C) COPYRIGHT IBM CORP. 19xx       '
   ```

   where '19xx' is the year of first availability for modules new in this Version/Release.

The program number generated by the macro has to be updated with every new version of VSE/POWER. The macro should be placed into the first 300 bytes of every VSE/POWER module. Care must be taken, that the module has established base register addressability already.

No registers are destroyed by the macro.

```
        IPW$CPY YB=nnnn,[YC=mmmm][BRANCH=YES|NO]
```

YB specifies the year of birth (general availability) of the subject module. This operand is mandatory.

YC specifies the year of latest change (general availability) of the subject module. This operand should not be used for modules, which are new in the current Version/Release.

BRANCH=YES|NO specify 'NO', if no branch instruction should be generated to bypass the copyright constant. This may be desirable, if the copyright constant is placed within a constant area. Specify 'YES', if the bypass branch instruction should be generated. 'YES' is the default.

IPW$CNC - Cancel VSE/POWER or VSE/POWER Task

The IPW$CNC macro is used to cancel either VSE/POWER or a VSE/POWER task. The macro expands into a linkage to the VSE/POWER task terminator (IPW$$TR) when a task is to be terminated or in a linkage to the VSE/POWER AB-exit when VSE/POWER is to be abnormally terminated.

```
[name]    IPW$CNC  CANCEL[,TYPE=POWER],PHASE=addr|(reg)

                            or

[name]    IPW$CNC  TYPE=TASK
```

CANCEL specifies to pass a cancel code of 255 (x'FF') to the AB-exit when abnormally terminated VSE/POWER. This parameter is applicable only when terminating VSE/POWER.

TYPE specifies the scope of termination

POWER causes to cancel the VSE/POWER partition by invoking the AB-exit. If this parameter is not specified, it is taken as default.

TASK causes to invoke the VSE/POWER task terminator routine in order to perform clean-up processing for the task concerned.

PHASE specifies the storage descriptor of the module causing the cancellation. If register notation is used, the designated register must have been previously loaded with the address of the module storage descriptor. The parameter is required when canceling VSE/POWER.

IPW$CTT - Perform Tape Control Operation

The IPW$CTT macro is used to perform a tape control operation, such as rewind or forward space file. The task must be equipped with a tape control block (TBB), which is pointed to by the TCB of the requesting task.

Registers 0 - 2 are destroyed by execution of the macro instruction.

```
[name]    IPW$CTT  {WTM|SNS|BSF|FSF|BSR|FSR|REW|RUN|SET|NOP}
```

BSF specifies to perform a 'backspace file' operation.

BSR specifies to backspace one record.

FSF specifies to perform a 'forward space file' operation.

FSR specifies to forward space one record.

NOP specifies to force device selection by a NOP req., CCW length to be 1

REW specifies to rewind the tape.

RUN specifies to rewind and unload the tape.

SNS specifies to sense the tape.

WTM specifies to write a tape mark.

Register 1 must have been previously loaded with the appropriate CCW length.

IPW$DQS - Delete Queue Entry from Class Chain

The IPW$DQS macro is used to delete a queue entry, pointed to by the TCB of the task, from the class chain if the disposition of the queue entry is 'D' or 'H'. If, however, the disposition is 'K', the queue entry is re-queued with leave ('L') disposition.

Note: The specified queue entry must have been previously obtained with a IPW$GQS macro instruction or otherwise properly addressed, else the action of the IPW$DQS macro call is unpredictable.

Registers 14 and 15 are used as linkage registers.

```
[name]   IPW$DQS   [HOLD|LOCATE][,LOCK=YES|NO]
```

HOLD causes to return the queue entry with its original disposition to its class chain for later processing; no eligibility posting is performed.

LOCATE causes to unchain the queue entry from the class chain it belonged to.

LOCK YES specifies that the delete queue entry routine performs a DMB release operation after processing the requested function; this is the default.

NO means that the calling task already owns the DMB and that the DMB should not be released.

IPW$DET - Detach VSE/POWER Task

The IPW$DET macro is used to detach the requesting task. The TCB is removed from the task selection list and the storage occupied by the TCB is returned to the VSE/POWER storage pool. The control is then given to the first eligible task in the task selection list.

```
[name]    IPW$DET  [RB][,ecb-addr]
```

ecb-addr specifies the address of an 4-bytes field, used as ECB, being posted when the task has been detached.

IPW$DSD - Define Storage Descriptor

The IPW$DSD macro is used to define a storage descriptor heading each VSE/POWER phase or control block. In most cases, the macro is used to generate a 16-bytes constant, containing the name and either the compile date of the phase or the last applied APAR.

```
[name]    IPW$DSD  [SECT=symbol][,REL=BASE|PNET|RJE][,APAR=number]
```

APAR specifies the name of the last integrated APAR. The parameter is not used during development.

REL BASE/PNET/RJE specifies that the phase belongs to the base product. The parameter must be coded when defining the storage descriptor of a VSE/POWER phase.

SECT specifies the name of the control block. The parameter need not be coded when defining the storage descriptor of a VSE/POWER phase. The macro uses the CSECT name of the phase as section name.

IPW$FQS - Free Queue Entry

The IPW$FQS macro is used to free the queue entry, pointed to by the TCB of the requesting task, and to return the DBLK group(s) occupied by the queue entry to one of the free DBLK group subchains, unless the queue entry is being accessed by another sas browse task (then it is added to the deletion queue if not already done so). Otherwise it is freed if:

- the disposition of the queue entry is either 'D' nor 'K'
- the disposition of the queue entry is either 'H' nor 'L' and the caller is the command processor
- the queue entry is in the deletion queue

Note: The queue entry must have been previously removed from the appropriate class chain by means of the IPW$DQS macro instruction.

Registers 14 and 15 are used as linkage registers.

```
[name]   IPW$FQS   LOCK=YES|NO]
```

LOCK YES specifies that the free queue entry routine performs a DMB release operation after processing the requested function; this is the default.

NO means that the calling task already owns the DMB and that the DMB should not be released.

IPW$GAM - Get Message and Send to Designated Person

The IPW$GAM macro is used to

- Write a message to the system or any remote operator
- Return the address of a particular message in the message definition module
- Copy the message into the caller's supplied area.

Registers 0 - 3 are destroyed by execution of the macro.

```
[name]    IPW$GAM  MSG=$mmmmm|(reg)
                   ,DEST=LOCAL|REMOTE|RETURN|address|(reg)
                  [,DESTID=(reg)]
                  [,REQ=ADDR[,RTDC=YES|NO]]

[name]    IPW$GAM  MSG=$mmmmm|(reg)
                   ,DEST=address|(reg),SUB=YES

[name]    IPW$GAM  MSG=0
                   ,DEST=address|(reg),SUB=YES

[name]    IPW$GAM  DOM=(R1)
```

MSG specifies the message identifier including its suffix; $mmmmm is the actual message number as defined in the message definition module, preceded with '$'. If register notation is used, the designated register must have been previously loaded with the message number.
If MSG=0 is specified, then this indicates only message substitution is to be done (requires SUB=YES and DEST=address|(reg)).

DEST specifies the destination of the message

LOCAL specifies to send the message to the system operator.

REMOTE specifies to send the message to the remote operator addressed via the DESTID parameter.

RETURN specifies to return the address of the message in register 1.

address specifies the address of an area to hold the copy of the message. The area must be large enough to accommodate the message text. If register notation is used, the designated register must have been previously loaded with the address of the area. If SUB=YES then an area of 132 bytes is required.

REQ ADDR specifies to return the address of the message in register 1. Upon return, register 0 is destroyed.

This macro is primarily used by VSE/POWER routines, which are not controlled by the VSE/POWER dispatcher, such as VTAM exits or subtasks.

DESTID specifies the target remote id. The parameter is only applicable when DEST=REMOTE is specified. If register notation is used, the designated register must have been previously loaded with the binary remote id number in its low order byte.

SUB=YES specifies that the message is to be fetched and message substitution is to be performed. It is allowed to use Registers 14 and 15 to contain data for message substitution. This operand requires DEST=address|(reg). An area of 132 bytes is required.

DOM=(R1) specifies that the message whose message ID is contained in register 1 is to be deleted from the console screen via the DOM macro. The message ID was obtained earlier from the TCB field TCMID after being issued by the IPW$GAM DEST=LOCAL or IPW$WTO macros.

RTDC=YES|NO Allowed only with the REQ=ADDR operand and requires a base register for the task control block(TCB). It specifies that the TCB is to be initialized with the message routing and descriptor codes as given in the IPW$GMM message description for passing on to the WTO (WTOR) macro when the message is issued.

IPW$GAR - Get Account Record

The IPW$GAR macro is used to get the first/next record from the account file. The account file must have been previously prepared for read operation by means of the IPW$OAF macro instruction.

Upon return, register 0 contains the length of the account record or is zero when an EOF record has been encountered. Register 1 contains the address of the account record.

Registers 14 and 15 are used as linkage registers.

```
[name]    IPW$GAR
```

IPW$GDR - Get Data Record

The IPW$GDR macro is used to get the first/next logical record from spool. Upon return, the record control word contained in the TCB of the requesting task contains the address, length and associated flags of the record.

Registers 14 and 15 are used as linkage registers.

```
[name]    IPW$GDR
```

IPW$GLR - Get Logical Record

The IPW$GLR macro is used to get the next record over the logical interface from the counterpart routine. Upon return, register 0 contains the address of the record and register 1 contains its length.

Register 14 is used as linkage register.

```
[name]    IPW$GLR  [RTN=LR|LO]
```

RTN LR or LO specifies that the logical routine acquires the next record from the physical routine.

If the parameter is omitted, the macro assumes that a physical routine acquires the next record from its corresponding logical routine.

IPW$GMS - Call General Message Service

The IPW$GMS macro is used to invoke the general message service function to perform one of the following:

- to route message and commands to the correct node and user,
- to perform message substitution,
- to remove two or more contiguous blanks from the message text.

Registers 14 and 15 are used as linkage registers.

```
[name]    IPW$GMS  TYPE=SUB|SQUEEZE

                   or

[name]    IPW$GMS  TYPE=DIST[,NMR=(reg)][,INTREC=(reg)]
```

TYPE specifies the type of function to be performed.

SUB causes to perform message modification. All message modification characters, contained in the message addressed by register 1 are replaced by the appropriate variables defined by the modification character. No other parameter is applicable.

Following registers must have been set up before execution of the macro instruction.

R0 message length - 1.
R1 address of message.
R4 address of local/remote message control block.
R5 TCB address to be used for message modification.

SQUEEZE causes to compress the passed message text whenever two or more contiguous blanks are found. No other parameter is required.

Following registers must have been set up before execution of the macro instruction.

R0 message length.
R1 address of message.

Upon return, register 0 contains the new (reduced) message length.

DIST causes to route the message or command to the designated node/user. Depending on the addressee, the message is sent to:

- Local system operator
- Any remote terminal operator
- Any VSE/ICCF user
- Any other SAS, assumed that a 'notify' communication path exists to that user.
- remote system (node)

NMR specifies the address of the nodal message record to be distributed. Register 1 is used to pass the address NMR record to the message service routine.

INTREC specifies that the message is in an internal format. Register 1 is used to pass the address of the internal message record to the message service routine.

IPW$GQR - Get Queue Record

The IPW$GQR macro is used to obtain the queue record, addressed by the I/O request word, from the storage copy of the queue file. Register 1 must point to the I/O request word.

Registers 0 - 3 are destroyed by the execution of the macro instruction.

```
 [name]   IPW$GQR   {address|(reg)}
```

address specifies the address of the I/O request word containing the relative queue record number and the address of the storage area to accommodate the queue record in question.

(reg) specifies that the address of the I/O request word is contained in the designated register.

Note: Length in "I/O request word" need not be set, Q-F-server always operates at length of Q-record.

IPW$GQS - Get Next Queue Entry

The IPW$GQS macro is used to obtain the next eligible queue entry from the specified class chains, contained in the task class list of the TCB, and to place the queue record of the queue entry in the queue record work area, pointed to by the TCB.

Upon return, the address of the queue record work area (TCQV) is zero if no queue entry is eligible. Furthermore, one of the following return codes is set in the function return code field of the TCB:

- No queue entry found
- Queue entry protected (password mismatch)
- Queue entry marked active
- Queue entry not in dispatchable state

Registers 14 and 15 are used as linkage registers.

```
[name]    IPW$GQS   [LOCK=YES|NO]
```

LOCK If "YES" is specified (the default value) then the DMB is released after processing the requested function. Otherwise it is assumed that the caller already owns the DMB and it should not be released.

IPW$GSL - GET SLI Record

The IPW$GSL macro is used to establish a linkage to the SLI processing routine (IPW$$SL) in order to perform one of the following functions:

- Locate specified member in the source statement library
- Get next member record
- Disconnect member / terminate SLI processing.

Registers 14 and 15 are used as linkage registers; register 0 is destroyed by execution of the macro instruction.

```
[name]    IPW$GSL  {FIND|GETR|PURGE}
```

FIND specifies to locate the member, specified in the parameter list, addressed by register 1.

GETR specifies to get the next record from the SLI member. Upon return, register 0 addresses the record and register 1 contains its length.

PURGE causes to terminate processing of the current SLI member or to terminate the entire SLI processing for the partition concerned when the termination code 'S' is set in the TCB.

IPW$GTE - Get Trace Entry

The IPW$GTE macro is used to allocate a trace area entry of the specified length in the active trace area. Upon return, register 1 contains the address of the trace area entry and register 0 contains its length.

Registers 2 and 3 are destroyed by execution of the macro instruction.

Note: The macro instruction can only be issued when either RJE or PNET is generated.

```
[name]    IPW$GTE  LENGTH=nnn|(reg)
```

LENGTH nnn specifies the length in bytes of the trace area entry to be allocated. If register notation is used, the designated register must have been previously loaded with the length.

IPW$GTO - Issue TD-Subtask Message

This access macro is for the message support of the TD-subtask.

It allows the caller to specify the message equate "msgid" of a message defined by the IPW$GMM macro in the IPW$$MM module. The message will be issued in the same way as for the maintask message support (IPW$GAM), using the WTO macro and providing message substitution and message squeezing via the IPW$$MX module. The message is issued to the console, and if the PSTART CNSLTR command has been issued specifying that internal tracing is to be performed with tracing message output being directed to SYSLST, then the message will be additionally issued to SYSLST.

```
[name]    IPW$GTO  MSG=msgid

[name]    IPW$GTO  MSG=TRACE

[name]    IPW$GTO  DOM=(R1)
```

MSG specifies the message identifier including its suffix; $mmmmm is the actual message number as defined in the message definition module, preceded with '$'. The WTO message id is returned to the caller in register 1 for later use in issuing the DOM macro with the IPW$GTO DOM=(R1) interface macro.

If MSG=TRACE is specified, this allows the caller to issue a PNET Driver Subtask trace message (1RTTI). The caller specifies in register 2 the length and in register 0 the address of a message containing the message number. The message is issued as is, without modification, either to the console or to SYSLST depending on the command PSTART CNSLTR.

DOM This access macro allows the caller to delete a console message issued previously by the IPW$GTO MSG= macro. The caller loads register 1 with the WTO message id returned by IPW$GTO MSG=.

IPW$GTS - Issue SD-Subtask Message

This access macro is for the message support of the SD-subtask.

It allows the caller to specify the message equate "msgid" of a message defined by the IPW$GMM macro in the IPW$$MM module. The message will be issued in the same way as for the maintask message support (IPW$GAM), using the WTO macro and providing message substitution and message squeezing via the IPW$$MX module. The message is issued to the console, and if the PSTART CNSLTR command has been issued specifying that internal tracing is to be performed with tracing message output being directed to SYSLST, then the message will be additionally issued to SYSLST.

```
[name]    IPW$GTS  MSG=msgid

[name]    IPW$GTS  MSG=TRACE

[name]    IPW$GTS  DOM=(R1)
```

MSG specifies the message identifier including its suffix; $mmmmm is the actual message number as defined in the message definition module, preceded with '$'. The WTO message id is returned to the caller in register 1 for later use in issuing the DOM macro with the IPW$GTS DOM=(R1) interface macro.

If MSG=TRACE is specified, this allows the caller to issue a PNET Driver Subtask trace message (1RTTI). The caller specifies in register 2 the length and in register 0 the address of a message containing the message number. The message is issued as is, without modification, either to the console or to SYSLST depending on the command PSTART CNSLTR.

DOM This access macro allows the caller to delete a console message issued previously by the IPW$GTS MSG= macro. The caller loads register 1 with the WTO message id returned by IPW$GTS MSG=.

IPW$IAS - Invoke Asynchronous Service

The IPW$IAS macro is used to attach, detach or request service from a VSE/POWER subtask. The following subtasks exist:

- Asynchronous service subtask
- Dump subtask
- Librarian subtask

Registers 14 and 15 are used as linkage registers; register 0 is destroyed by execution of the macro instruction.

```
[name]    IPW$IAS  TYPE=ATTACH|DETACH|SERVICE[,TASK=DUMP|LIBR]
```

TYPE specifies the type of request to be performed.

ATTACH causes to attach the subtask described by the TASK parameter.

DETACH causes to detach the subtask described by the TASK parameter.

SERVICE causes to pass a 'service' request to the associated subtask. Register 1 must point to a service request block (SRB), describing the requested service.

TASK specifies the name of the subtask. If the parameter is omitted, the asynchronous service subtask is assumed.

DUMP specifies that the request is for the dump subtask.

LIBR specifies that the request is for the librarian subtask.

IPW$ICP - Invoke VSE/POWER Command Processor

The IPW$ICP macro is used to pass a command to the VSE/POWER command processor. A temporary command processor task is built and attached.

Register 0 must either address an ECB, which is posted when the command is processed, or be zero.

Registers 14 and 15 are used as linkage registers.

```
[name]    IPW$ICP  [NMR=YES|NO] [,REQ=POWER]
             [,PASS=PCE|VDCLT|NOLOCK|DCK|QEN|INIT][,WAIT=NO]
```

NMR specifies whether the command, addressed by register 1, is in nodal message record (NMR) format or not.

YES specifies that the command is in NMR format. The length of the command is contained in the NMR header.

NO specifies that register 1 points to a 72-bytes area containing the 'free format' command. The command must be passed in uppercase characters. NO is the default if the parameter is omitted.

REQ=POWER specifies that for the passed command no authority check will be done.

PASS specifies that the contents of register 2 should be passed to the temporary command processor task within TCB field 'CPPA', and that its logical meaning is expressed by an indication within TCB flag 'CPFG' or 'CPFG2'.
For certain values the operand merely passes an informational flag with register 2 not respected.

PCE specifies that the passed field should be interpreted (see 'CPFG') as the Partition Control Block Extension (PCE) address of a dynamic partition to be PSTART'ed.

VDCLT specifies that the passed field should be interpreted (see 'CPFG') as a pointer to the $RSV dynamic class table area, which amongst others contains the verified dynamic class table.

NOLOCK specifies that the invoked command processor task should not lock/unlock the DPCB (see 'CPFG'); contents of register 2 is not passed.

DCK specifies that the invoked command processor task should just delete the checkpoint information for the passed queue entry (see 'CPFG').

QEN specifies that the passed field should be interpreted (see 'CPFG2') as the internal queue record number of the queue entry to be addressed by the command.

INIT specifies that the invoked command processor task should allow a node name change.

The macro IPW$ICP propagates the PASS= indication to the new TCFRB2 field in the TCB.

WAIT=NO This option sets the TCF8 flag TCF8NW. It specifies not to set the task into wait state, when no real/PFIXED storage is currently available to create a temp. cmd. processor TCB. Instead control will return to the caller with TCF8NW still set. The caller should test TCF8NW, clear it on its own, and take corrective action.
When real/pfixed storage is available to create a temp. cmd. TCB, TCF8NW is reset by IPW$$IC before return to the caller.
NOTE: The default WAIT= option is 'YES', and TCF8NW is unconditionally reset to zero. The option means, the calling task should be put into wait state, if no storage is available.

NOTE: When $ICP is called both with WAIT=NO and R0¬=0, to wait for the final completion of the invoked command, then TCF8NW setting is propagated into the temp.cmd.proc. TCB, which in turn may then avoid $RSW/$RSV storage waits but signal shortage to the caller in the R0-ECB post byte by

- X'02' (use EQU NOREAL), in case $RSW failed
- X'01' (use EQU NOVIRT), in case $RSV failed.

The X'80' posted caller should check and clear the same byte for additional '02'/'01' settings and take action.

IPW$ICS - Invoke Common Services

The IPW$ICS macro is used to add, delete or obtain a message or command, in NMR format, from the message/command queue anchored in the NCB.

Registers 0 - 3 are destroyed by execution of the macro.

```
[name]    IPW$ICS  REQ=ADD|GET|DEL[,NMR=(reg)][,TCB=(reg)]
```

REQ specifies the request to be performed.

ADD causes a message or command in NMR format addressed by the register specified in the NMR parameter or contained in register 1, if the NMR parameter is omitted, to be added as the last entry of the message/command queue.

DEL causes the storage occupied by the nodal message to be returned to the VSE/POWER storage pool. If the NMR parameter is not coded, register 1 must contain the address of the NMR to be deleted.

GET specifies to return the address of the first record in the message/command queue in register 1. The message or command will be unchained from the queue. If there are no entries in the queue then register 1 will contain zero. Register 1 must have been previously loaded with the address of the associated node control block (NCB).

NMR specifies the address of the nodal message or command to be added or deleted from the appropriate queue.

TCB specifies the address of the TCB which should be used for message modification. If not specified then the own TCB will be used. If a register other than register 5 is used then the contents will be loaded into register 5 and its contents will be destroyed on exit.

IPW$IDM - Invoke IDUMP of the VSE/POWER Partition

The IPW$IDM macro is used to call the IDUMP processor module IPW$$ID. For details refer to the VSE/POWER "Administration and Operation" Guide, "Appendix B. VSE/POWER Diagnostics and Service Aids".

IPW$IDS - Invoke Data Management Service Routines

The IPW$IDS macro is used to establish a linkage to the data management service routines to perform one of the following functions:

- Position spooling pointers at specified record (restart).
- Spool and update record/line and page counts.
- Set EOD flag in last spooled record.
- Replace data set header record on spool by new data set header record of same length.
- Replace job header record on spool by new job header record of same length.
- Adjust page-count-increment for CDPS record.

Registers 14 and 15 are used as linkage registers.

```
[name]    IPW$IDS  REQ=RESTART|SETEOD|SPOOL|REPLDSHR|REPLJHR|COUNTPG[,LOCK=YES|NO]
```

REQ specifies the request to be performed.

COUNTPG causes the IPW$$DS routine to be called to adjust the caller's passed page count increment depending on the current page-count-state and the current record to be processed.

RESTART causes to restart the spooling process and resetting the spooling pointers to the specified record. Register 1 must contain the record number + 1 from where to restart.

SPOOL causes to check the carriage control character associated with the data record (TCCC) and optionally to spool the record.

SETEOD causes to set the EOD indicator to the last written data record for the queue entry addressed by the TCB.

REPLDSHR causes to replace the data set header record on spool by the data set header record, addressed by the record control word of the calling task.

REPLJHR causes to replace the job header record on spool by the job header record addressed by the record control word of the calling task.

COUNTPG causes the pre-evaluated TCG4.TCIPC 'page increment indication' to be adjusted for CPDS records according to the current page count state and the current CPDS record type.

LOCK YES specifies that the called routine performs a DMB release operation after processing the requested function; this is the default.

NO specifies that the calling routine already owns the DMB and that the DMB should not be released. Only applicable for REQ=RESTART.

IPW$IIS - Invoke Print Status Processing Service

The IPW$IIS macro calls the print status service module IPW$$PS1 to perform the eligibility checking of a queue record according to the criteria in a print status work area passed by the caller.

Register 1 points to the queue record to be checked.

Register 2 points to the print status work area (header) containing the checking criteria.

Registers 14 and 15 are used as linkage registers.

```
[name]    IPW$IIS   [REQ=DIREL]
```

REQ specifies (setting TCFRB2) non standard requests to be performed.

DIREL causes to address the 'direct' instead of 'normal' eligibility checking routine of IPW$$PS1 to be entered, and expects register 4 to point to the XT-workarea of the calling SAS task.

IPW$IOM - Invoke I/O Monitor or SNA Send/Receive Routine

The IPW$IOM macro is used to call the RJE BSC, PNET BSC or PNET SNA I/O manager.

Registers 14 and 15 are used as linkage registers.

```
[name]    IPW$IOM   [TYPE=SEND|SENDX|RECEIVE|RECEIVEX][,NCB=(reg)]
```

NCB specifies the address of the node control block (NCB) to be used for the I/O operation. The parameter is only applicable for the PNET BSC or SNA I/O manager.

TYPE specifies the type of PNET,SNA I/O operation to be performed.

SEND causes to perform de-queueing of the 'head' SNA output buffer from the 'to-be-sent' queue and to send the buffer by means of the SEND macro instruction.

RECEIVE causes to de-queue the first buffer from the free input queue and to issue a VTAM RECEIVE macro.

SENDX causes to free the RPL by means of the CHECK macro. The macro is only applicable for the SEND exit. The buffer is queued at the head of the 'channel end' queue.

RECEIVEX causes to free the RPL by means of the CHECK macro. The macro is only applicable for the RECEIVE exit. The input buffer is queued at the tail of the received input buffer queue.

IPW$IOC - Invoke Compaction Processing

The IPW$IOC macro is used to load a new compaction table in the VSE/POWER GETVIS area, if the compaction table is not yet present. The macro expands into a linkage to the compaction processing routine IPW$$OC.

Registers 14 and 15 are used as linkage registers.

```
[name]    IPW$IOC
```

IPW$IPS - Invoke PNET Driver Routines

The IPW$IPS macro provides an interface between the PNET driver modules (IPW$$LDn). The IPW$IPS macro also provides an interface between the PNET receiver modules (IPW$$NRn).

Registers 14 and 15 are used as linkage registers.

```
[name]    IPW$IPS  [MOD=LD1|LD2|LD3|LD4|NR2]
                   [,FCT=RIFRCB|TRNRCB|RCVRCB|LOGERR|CNLTSK|
                         CRETSK|UPDNAT|SETTIM]
```

MOD specifies a short name of the module to be called.

FCT specifies the function to be provided by a subroutine which is located in IPW$$LD5.

RIFRCB causes to validate the SRCB of a buffer containing an RIF.

TRNRCB causes to validate the SRCB of a buffer containing an MLI-control record for a transmitter.

RCVRCB causes to validate the SRCB of a buffer containing an MLI-control record for a receiver.

LOGERR causes to write an error-record onto the SYSREC file using the SVC 44.

CNLTSK causes to propagate a stop-code for transmitters and/or receivers.

CRETSK causes to create a VSE/POWER subtask to initiate or complete a SNA session.

UPDNAT causes to update the NAT, i.e. delete a node.

SETTIM causes to set up a variable timer interval of n/10th seconds, as passed in field PASTIME.

Notes:

1. Register R6 must point to the NCB. The fields PNCBALDn and PNCBDS are referenced.
2. If FCT specified, the field NCBFCT5 is changed and the field NCBDS is referenced. Depending on the function, any parameters up to the length of 12 bytes may be passed in NCBPAR5 to IPW$$LD5 or returned by IPW$$LD5.

IPW$IQS - Invoke Queue Management Service Routines

The IPW$IQS macro is used to establish a linkage to the queue management service routines (IPW$$SQ and IPW$$Q1).

Registers 14 and 15 are used as linkage registers.

```
[name]   IPW$IQS   REQ=ALLOCGP|FREEGPS[,LOCK=YES|NO]

                                 or

[name]   IPW$IQS   REQ=FORMAT|CLTAB

                                 or

[name]   IPW$IQS   REQ=BUILDSLOT|DELSLOT|POSTSLOT|PROCSLOT|CLEARSLOT|READSLOT|FREEQCA
                   [,TYPE=NMR|WFW|CKP]
                   [,SYSID=sysid]
```

REQ defines the type of request to be performed by the queue service routines or the slot manager, which is part of the queue service routines.

CLTAB causes to build the class table pointers in the class list, contained in the task control block (TCB) of the task concerned. Up to 4 classes can be specified. The classes must be specified in the high-order byte of the 4-bytes class table pointer field. The end of the class list is indicated by X'FF'. Register 1 must contain the address of the TCB for which the class table pointers should be built. Register 0 must contain in its low order byte, the queue type, either 'L' for LST queue, 'P' for PUN queue or 'R' for RDR queue.

BUILDSLOT causes to construct either a waiting for work slot, if TYPE=WFW is specified or a message/command slot, if TYPE=NMR is specified or a checkpoint slot, if TYPE=CKP is specified and to place the slot in the shared queue control area. Register 1 contains the address of the external device control block (EDCB), if WFW slot to build or the address of the nodal message record, if a NMR slot to build, or the address of the checkpoint control record, if a CKP slot to build. The macro expansion sets up register 0 with the slot type identifier.

DELSLOT causes to delete a waiting for work slot, named by the external device name or a checkpoint slot, identified by the queue record from the shared queue control area. Register 1 addresses the external device control block concerned.

POSTSLOT Causes to scan the shared queue control area in order to post all waiting for work slots which can process the queue entry just added in the class chain. Register 1 addresses the queue record of the queue entry added.

PROCSLOT causes to scan the shared queue control area in order to process all message/command slots destined for the local system and all posted waiting for work slots owned by the local system.

CLEARSLOT causes to remove all slots in the shared queue control area which are either destined to or owned by the system(s) to be recovered.

READSLOT causes to retrieve a checkpoint slot from the queue control area. The checkpoint slot is selected according the jobname, jobnumber, jobsuffix, and queue id specified in the queue record addressed by TCBQV, and the record number specified in TCBQW.

FREEQCA causes to free the entire QCA after an I/O error.

ALLOCGP causes to unchain the first DBLK group from one of the free DBLK group subchains and to return the relative DBLK number of the first DBLK in that DBLK group in register 1.

FREEGPS causes to return the DBLK group(s) on top of one of the free DBLK group subchains. Register 1 must contain the relative DBLK number of the first DBLK in the first DBLK group, register 2 must contain the relative DBLK number of the first DBLK in the last DBLK group and register 3 must contain the number of DBLK groups to be released.

FORMAT causes to format the queue file, if the file resides on CKD device. All device specific information are extracted from the queue file MCB and the DMB.

Upon return, register 15 contains 0 if the formatting was successful. Register 15 contains 4 if the queue file formatting failed.

LOCK YES specifies that the called routine performs a DMB release operation after processing the requested function; this is the default.

NO means that the calling task already owns the DMB and that the DMB should not be released.

TYPE defines the type of slot to be built or deleted.

NMR indicates message/command slot.

WFW indicates 'waiting for work' slot.

CKP indicates checkpoint slot.

SYSID specifies the name of a one-byte field containing the node qualifier (=System Id) of the local target system, to which - for example -, the nodal message record is to be routed.

IPW$IRY - Invoke Queue File / Account File Recovery

The IPW$IRY macro is used to establish a linkage to the queue/account file recovery routines.

Registers 14 and 15 are used as linkage registers.

```
 [name]   IPW$IRY   REQ=QUEUE|ACCOUNT[,PARM=address|(reg)]
```

REQ specifies the type of recovery action to be done.

QUEUE specifies that queue file recovery is performed. In a shared spooling environment register 0 must address a parameter list containing the 1-byte SYSIDs of the systems to be recovered. The parameter list must be delimited by X'FF'.

ACCOUNT specifies that account file recovery is performed. The EOF record on the account file is located and the remaining capacity is calculated and saved in the account control block.

PARM specifies the parameter list containing the 1-byte SYSIDs of the systems to be recovered (shared spooling only). Register 0 is used to point to the parameter list.

address specifies the address of the parameter list.

(reg) specifies that the address of the parameter list is contained in the designated register.

IPW$ITP - TD-Subtask EZASMI Interface

This access macro is to support the TD-subtask for issuing EZASMI API socket call requests.

The caller specifies via the PARMS= operand the EZASMI socketcall desired. The return code is found in the field TPWTPR1.

```
[name]    IPW$ITP  PARMS=(R1)

[name]    IPW$ITP  PARMS=(R1),CKRC=YES
```

PARMS Using the IPW$ITP macro, the subtask may invoke the EZASMI API for the following socketcalls. The socketcall request is speicifed in register 1 with the socketcall value:

- 05 = ACCEPT
- 13 = BIND
- 09 = CANCEL
- 08 = CLOSE
- 14 = CONNECT
- 01 = INITAPI

- 03 = GETHOSTID
- 10 = GETHOSTBYADDR
- 11 = GETHOSTBYNAME
- 04 = LISTEN
- 07 = RECEIVE
- 06 = SEND
- 12 = SOCKET
- 02 = TERMAPI

The EZASMI interface is invoked in 31-bit mode. Internally the IPW$$TS module will invoke the IPW$ITP CKRC=YES macro (see below) to check the EZASMI socketcall for any immediate error return. If required, an IDUMP may be taken for the individual error situation and connection.

CKRC This macro checks for errors returned by the EZASMI API. Besides setting the final return code in the the field TPWTPR1, it also issues the IDUMP macro if needed for any given error return code. The caller is notified of the final result with either of the return codes:

- 0 = OK (TPWTPR10)
- 4 = RETRY REQUEST (TPWTPR14)
- 8 = DISCONNECT LINE (TPWTPR18)
- 12 = SHUTDOWN TCPIP (TPWTPR1C)

IPW$ITS - SD-Subtask EZASMI Interface

This access macro is to support the SD-subtask for issuing EZASMI API TCP/IP SSL call requests.

The caller specifies via the PARMS= operand the EZASMI socketcall desired. The return code is found in the field TPWTPR1.

```
[name]    IPW$ITS  PARMS=(R1)

[name]    IPW$ITS  PARMS=(R1),CKRC=YES
```

PARMS Using the IPW$ITS macro, the subtask may invoke the EZASMI API for the following socket and SSL calls. The request is speicifed in register 1 for the call value:

- 05 = ACCEPT
- 13 = BIND
- 09 = CANCEL
- 08 = CLOSE
- 14 = CONNECT
- 03 = GETHOSTID
- 10 = GETHOSTBYADDR
- 11 = GETHOSTBYNAME
- 17 = GSKINIT
- 18 = GSKUNINIT
- 19 = GSKGETDNBYLAB
- 20 = GSKFREEMEM
- 21 = GSKSSOCINIT
- 22 = GSKSSOCREAD
- 23 = GSKSSOCWRITE
- 24 = GSKSSOCCLOSE
- 25 = GSKSSOCRESET

- 26 = GSKGETCIPHINF
- 01 = INITAPI
- 27 = IOCTL
- 04 = LISTEN
- 07 = RECEIVE
- 15 = SELECT(Read)
- 16 = SELECT(Write)
- 06 = SEND
- 12 = SOCKET
- 02 = TERMAPI

The EZASMI interface is invoked in 31-bit mode. Internally the IPW$$SS module will invoke the IPW$ITS CKRC=YES macro (see below) to check the EZASMI call for any immediate error return. If required, an IDUMP may be taken for the individual error situation and connection.

CKRC This macro checks for errors returned by the EZASMI API. Besides setting the final return code in the the field TPWTPR1, it also issues the IDUMP macro if needed for any given error return code. The caller is notified of the final result with either of the return codes:

- 0 = OK (TPWTPR10)
- 4 = RETRY REQUEST (TPWTPR14)
- 8 = DISCONNECT LINE (TPWTPR18)
- 12 = SHUTDOWN TCPIP (TPWTPR1C)

IPW$ITQ - Invoke Maintain Wait for Run Subqueue

The IPW$ITQ macro is used to establish a linkage to the wait for run subqueue routines located in IPW$$TQ.

Registers 2 and 1 are used as linkage registers.

```
 [name]   IPW$ITQ   ADD|DEL|INIT
```

ADD specifies that IPW$$TQ calculates the due date for the queue entry and if necessary adds it to the wait for run subqueue.

DEL specifies that IPW$$TQ deletes the queue entry from the wait for run subqueue.

INIT specifies that IPW$$TQ scans the wait for run subqueue for queue entries with expired due dates. These queue entries are added to the really dispatchable chain.

IPW$IXS - Invoke Cross-Partition Services

The IPW$IXS macro provides an interface between the spool access interface routines (IPW$$XTn).

```
[name]    IPW$IXS  FCT=CTL|GET|PUT|SUB
                   [,REQ=WEV|RCV|REP|XPE|SMD|JHR|DSH]
```

FCT specifies the function to be provided.

CTL causes to process the CTL request by invoking the IPW$$XTC module.

GET causes to process the GET request by invoking the IPW$$XTG module.

PUT causes to process the PUT request by invoking the IPW$$XTP module.

SUB causes to process a request which is specified in the REQ parameter by invoking the IPW$$XTS module.

REQ specifies the request to be provided, if SUB has been specified for the FCT parameter.

WEV causes to wait till the next event has been posted, which may be either the posting of the Receive-ECB for the XPCC-support or the posting of the own task ECB.

RCV causes to issue the XPCC macro with FUNC=RECEIVE and do a testing of the return codes and prechecking of the user data (containing the action bytes, buffer values, etc.) and some of the SPL-parameters.

REP causes to issue the XPCC macro with FUNC=REPLY and do a testing of the return codes.

XPE causes to test the return codes after the XPCC macro has been issued with FUNC=RECEIVE or REPLY.

SMD causes to send a message for a device service task (DST). The receiver of the message is passed within the XTWAREA.

JHR causes to update the SPL (addressed by TCBXSPL) with the information out of the job header record (addressed by TCBRV) and queue record (address in TCBQV).

DSH causes to update the SPL (addressed by TCBXSPL) with the information out of the data set header record (addressed by TCBRV).

Notes:

1. The registers R14 and R15 are changed.
2. If FCT=SUB specified, register R1 and the field XTWSUB are changed and the field XTWSUBM1 is referenced, i.e. in PLS code the field XTWSUB should be specified in the SETS option and XTWSUBM1 in the REFS option in at least one GEN statement. The registers R1, R14 and R15 are changed.

IPW$MQR - Modify Queue Record

The IPW$MQR macro is used to modify the queue record, addressed by the I/O request word, in the storage copy of the queue file only. Register 1 must point to the I/O request word.

Registers 0 - 3 are destroyed by the execution of the macro instruction.

```
[name]   IPW$MQR   {address|(reg)}
```

address specifies the address of the I/O request word containing the relative queue record number and the address of the storage area which holds the new version of the queue record.

(reg) specifies that the address of the I/O request word is contained in the designated register.

Note: Length in "I/O request word" need not be set, Q-F-server always operates at length of Q-record.

IPW$NTY - Notify User

The IPW$NTY macro is used to send a message, which is not in the NMR message format, to either a remote operator attached to the own node, to a user on another node, to the central operator, or to an ICCF user.

The macro can also be used to add a message, already in NMR format or internal message record format, to the tail of the appropriate Notify message queue, or to add a message to a message queue for later retrieval by an application program via VSE/POWER's Spool-access support.

Registers 0 - 3 are destroyed by execution of the macro. If QCM=YES is specified, register 4 is destroyed too.

```
[name]     IPW$NTY  MSG=$nnnnn|(reg)[,USER=(reg)]
                   [,QUAL=qualifier]
                   [,QCM=YES]
                   {[,NODE=(reg)]|[,APPL=(reg)]}

                           or

[name]     IPW$NTY  {NMR=(reg)|INTREC=(reg)}
```

NMR   specifies the address of the nodal message record which must be queued in the appropriate Notify queue. If a register other than register 1 is specified then the NMR address will be loaded into register 1.

INTREC   specifies the address of an internal message record (VSE/POWER message order control record) to be queued at the tail of the appropriate subsystem 'notify' queue.

MSG $nnnnn is the actual message number as obtained from the message definition module prefixed with $. If register notation is used, the designated register must have been previously loaded with the message number.

NODE   specifies the address of an eight byte field containing the target node name. If not specified the local node is assumed to be the required destination.

QUAL   specifies the name of a one-byte field containing the node qualifier.

QCM   specifies that any generated job completion message is added to a fixed format message queue for later retrieval by an application program. Register 0..4 are destroyed after execution. Specification of USER, QUAL and NODE is required. This operand is only sensible for an execution reader, network receiver or timer task processing the macro.

USER   specifies the address of an eight byte field containing the target userid, either a remote id in the form 'Rnnn' or an ICCF, TSO, or CMS userid. If not specified, the message is routed to the local console of the specified node.

APPL   specifies the address of an eight byte field containing the subsystem name. The parameter is mutually exclusive with the NODE parameter.

IPW$OAF - Open Account File for Read Mode

The IPW$OAF macro is used to open the account file for read operation. The current write CCW-chain is modified into a read CCW-chain.

Registers 14 and 15 are used as linkage registers.

```
[name]    IPW$OAF
```

IPW$OEF - Open 3540 Diskette File

The IPW$OEF macro is used to open a 3540 diskette file. The task must be equipped with a 3540 physical work space pointed to by the TCB of the calling task. The 3540 physical work space contains device specific information, such as device address, and the name of the file to be opened. Upon return, the physical work space contains extent information and the record length.

Registers 14 and 15 are used as linkage registers.

```
[name]    IPW$OEF
```

IPW$OLI - Open Logical Interface

The IPW$OLI macro is used to open the interface to the logical reader (IPW$$LR), logical writer (IPW$$LW), or the logical output spooler (IPW$$LO). The macro expansion obtains a new register save area and saves the entry point address of the logical routine in the new save area.

Registers 0 - 3 are destroyed by execution of the macro.

```
[name]    IPW$OLI  [lrtn|(reg)]
```

lrtn specifies the name of the logical routine to be opened. If register notation is used, the address of the logical routine must have been loaded in the designated register before execution of this macro instruction; registers 0 - 3 cannot be used.

LR	logical reader (IPW$$LR)
LW	logical writer (IPW$$LW)
LO	logical output spooler (IPW$$LO)

IPW$OPI - Invoke Output Parameter Processing Routine

The macro IPW$OPI is used to invoke "user defined" output parameter processing routine.

Registers 14 and 15 are used as linkage registers. Register 1 contains the address of the passed parameter list.

```
[name]    IPW$OPI  FUNC=OPDEBLD|OPANAL|OPPUT|OPGET|OPMOD,
                   PARM=(reg)
```

FUNC specifies the function to be performed:

OPDEBLD parses the DEFINE statement pointed to by the parameter list, builds an Output Parameter Definition Entry (OPDE), if the DEFINE statement is valid and adds the built OPDE to the OPDE-chain.

OPANAL analyzes the user defined output parameter pointed to by the parameter list and builds an Output Parameter Text Block (OPTB), if the parameter corresponds to the definition in the appropriate OPDE and appends the OPTB to the output processing section of the data set header.

OPPUT analyses the output parameter text blocks in the code point area pointed to by the parameter list and appends the OPTB(s) to the output processing section of the data set header, if the (all) OPTB(s) is (are) valid.

OPGET Retrieves one specific OPTB or all OPTBs from the output processing section of the data set header.

OPMOD Modifies one specific OPTB in the output processing section of the data set header by replacing it with a new one, which must be equal in length to the old one.

PARM specifies the register, which contains the pointer to the required parameter list. The layout of the parameter list depends on the function to be performed.

IPW$OTP - Open Tape Processing

The IPW$OTP macro is used to create a tape control block (TBB) used for subsequent tape processing, open, or close tape processing. The macro expands into a linkage to the 'open tape' routine (IPW$$OT).

Registers 14 and 15 are used as linkage registers.

```
[name]    IPW$OTP  {BC|DB|OT|CT|FEOVC|FEOVN|FEOVS|MLAST|MVOL1},{RD|WR}
```

For the 1st positional operand:

BC specifies to build a tape control block and anchor the TBB to the TCB of the requesting task.

DB specifies to delete the tape control block

FEOVC specifies to perform a BAM volume change for a continued queue record

FEOVN specifies to perform a BAM volume change for a non-continued queue record

FEOVS specifies to perform a BAM volume change for a SYSIN tape

MLAST specifies to require the mounting of the last BAM tape volume for a given spool entry

MVOL1 specifies to require the mounting of the first BAM tape volume for a given spool entry

OT specifies to open the tape either in write or read mode.

CT specifies to close the tape.

For the 2nd positional operand:

WR specifies that the tape is being used in 'write' mode.

RD specifies that the tape is being used in 'read' mode.

IPW$PAR - Write Account Record

The IPW$PAR macro is used either to build and write an account record or just to write the account record to the account file.

Registers 14 and 15 are used as linkage registers; registers 0 and 1 are used as interface registers.

```
[name]    IPW$PAR  [REC=RDR|OUT|SPOOL]
```

REC specifies the type of account record to be built and written. If the parameter is omitted, the macro expands into a linkage to the 'write account record' routine (IPW$$PA/IPW$$PF). In this case, registers 0 and 1 must contain the length or address of the account record, respectively.

RDR specifies to build a reader account record bases on the information supplied by the calling task (queue record). If the calling task is a SAS task, a spool account record is built instead.

OUT specifies to build either a list or punch account record, depending on the information supplied by the calling task. A spool account record is built instead of a list/punch account record when the calling task is a SAS task.

SPOOL specifies to build a spool account record.

IPW$PDR - Put Data Record

The IPW$PDR macro is used to pass a logical record, described by the record control word in the TCB of the requesting task (TCRW), to data management for subsequent spooling. The record is buffered in a DBLK.

Registers 14 and 15 are used as linkage registers.

```
[name]    IPW$PDR
```

IPW$PLR - Put Logical Record

The IPW$PLR macro is used to pass a record over the logical interface to the counterpart routine. Register 0 must contain the address of the record and register 1 must contain its length.

Register 14 is used as linkage register.

```
[name]    IPW$PLR  [RTN=LW]
```

RTN LW specifies that the logical writer routine passes a record to the physical routine.

If the parameter is omitted, the macro assumes that a physical routine passes a record to its corresponding logical routine.

IPW$RDC - Get Time of Day (Read Clock)

The IPW$RDC macro obtains the time of day and updates the date field in the disk management block (DMB) with the current date. The time is returned in register 1 as a packed decimal quantity of the form 0HHMMSSC, where C is a 4-bit sign character that allows the time to be unpacked and printed.

Note: The time returned is the time of day based on a 24-hour clock.

Registers 0, 2 and 3 are destroyed by execution of the macro.

```
[name]    IPW$RDC
```

IPW$RDD - Read Data Block from Disk

The IPW$RDD macro is used to read the data block, addressed by the I/O request word, from the VSE/POWER data file. The I/O request word consists of a 12-bytes parameter list, containing the relative DBLK number of the DBLK to be read in and the address where to read the record in storage.

Registers 0 - 3 are destroyed by execution of the macro.

```
 [name]   IPW$RDD   {address|(R1)}[,IO=YES|NO][,WAIT=FORCE]
```

address specifies the address of the I/O request word used for the read operation.

(R1) specifies that the address of the I/O request word (12 bytes), used for the read operation, is contained in register 1.

IO specifies whether or not to perform the read operation.

YES specifies to perform the read operation; this is the default, if the parameter is omitted.

NO specifies not to perform the read operation. The parameter is used to ensure that the previous read for the DBLK addressed in the I/O request word has been completed (only applicable when using double data file buffering).

WAIT FORCE specifies that the disk service routine must wait for the I/O completion regardless of any 'double buffering' flag set.

Note: The I/O request word provided length field will be used, if specified - both for reading a DBLK or a SER record only; if specified = 0, then disk service operates with the constant "DBLK length".

IPW$RDQ - Read Queue Record from Disk

The IPW$RDQ macro is used to read the queue record block or master record, addressed by the I/O request word, from the VSE/POWER queue file.

Registers 0 - 3 are destroyed by execution of the macro.

```
[name]    IPW$RDQ  {address|(R1)}[,LOCK=YES|NO]
```

address specifies the address of the I/O request word used for the read operation.

(R1) specifies that the address of the I/O request word (12 bytes), used for the read operation, is contained in register 1.

LOCK YES specifies that the disk service routine, contained in the VSE/POWER nucleus performs a MCB release operation after processing the requested function; this is the default.

NO means that the calling task already owns the MCB and that the MCB will not be released at completion of the I/O.

Note: The I/O request word provided length field will be used, if the Q-record-block number is the block number of the master record; if not, then disk service will operate with the constant "queue record block length".

IPW$RDT - Read Tape Record

The IPW$RDT macro is used to read a record from tape, described by the tape control block (TBB), which is pointed to by the TCB of the requesting task.

Registers 0 - 2 are destroyed by execution of the macro instruction.

```
[name]    IPW$RDT  {TCQW|TCDW|(R1)}
```

TCQW specifies to read the queue record from tape. The queue record is read in the area, addressed by field TCQW.

TCDW specifies to read a data block (DBLK) from tape. The data block is read in the area, addressed by field TCDW.

(R1) specifies to read a record from tape into the area addressed by register 1. The length of the record must have been stored in the TBB before execution of the macro instruction.

IPW$RET - Return to Caller

The IPW$RET macro is used to restore the registers 14, 15, and 0 through 12 from the current save area, addressed by register 13, and to return to the calling routine by branching to the location addressed by register 14.

```
[name]    IPW$RET  [RETCODE=nn]
```

RETCODE nn specifies a numeric return code to be returned in register 15. If the operand is not specified, no return code is set.

IPW$RLR - Release Resource

The IPW$RLR macro is used to unlock a VSE/POWER resource (control block).

Register 2 is destroyed by execution of the macro instruction and register 3 contains the address of the control block concerned.

```
[name]    IPW$RLR  symbol|(reg)
```

symbol specifies the name of the control block to be released. This is the name of the field containing the address of the control block, as defined in the CAT, with the first two characters stripped off. If register notation is used, the designated register have been previously loaded with the address of the control block concerned.

IPW$RLV - Release GETVIS Storage

The IPW$RLV macro is used to release GETVIS storage previously acquired with the IPW$RSV macro instruction. The storage is returned to the appropriate pool and is available for other tasks. All tasks waiting for storage are again posted in an attempt to satisfy the requirements.

Registers 0 - 3 are destroyed by execution of the macro instruction.

```
[name]    IPW$RLV  {ALL|ADDR=(reg)}
                   [,OWNER=(reg)][,ANCHOR=(reg)]
                   [,LENGTH=nnnnn|(reg)][,PREFIX=YES|NO]
```

ADDR specifies the address of the GETVIS storage area to be released. The length of the storage area is found from the storage prefix. The parameter is not required when 'ALL' is specified. Register 0 can be used as parameter register.

ALL specifies to release all storage areas belonging to the task or to a particular storage chain.

OWNER specifies the address of the TCB of the task that is to be used as owner for this storage. If not specified the issuing task will be considered as owner. The 'head' pointer will be taken from the owners TCB. Must NOT be specified if ANCHOR is specified.

ANCHOR specifies the address of a double word containing the head and tail pointer of the storage chain. The parameter is mutually exclusive with the OWNER parameter.

PREFIX specifies whether or not the storage area to be returned is preceded by a VSE/POWER system prefix.

YES specifies that a prefix, containing the length of the storage precedes the storage area. This is the default when the parameter is omitted.

NO specifies that no prefix precedes the storage area. If 'NO' is specified, also the LENGTH parameter must be coded.

LENGTH specifies the length in bytes of the storage area to be returned to the GETVIS pool. The parameter is only applicable together with the PREFIX=NO specification. Register 1 cannot be used as parameter register.

IPW$RLW - Release Fixed (Real) Storage

The IPW$RLW macro is used to release a storage area, which was previously obtained by means of the IPW$RSW macro instruction, and return the storage to the VSE/POWER real storage pool.

All VSE/POWER tasks waiting for real storage are again be posted in an attempt to satisfy the requirements.

```
[name]    IPW$RLW  (R1)
```

The address of the storage to be freed must have been previously loaded in register 1. Upon return registers 0 and 1 contain hex. zero; registers 2 and 3 are destroyed.

IPW$RMS - Remote Message Service

The IPW$RMS macro is used to queue a message to the ALLUSERS queue or to the message queue associated with the remote id, to obtain the first/next message from the specified queue, or to delete a particular message from the remote message queue or ALLUSERS queue.

Registers 0 - 3 are destroyed by execution of the macro instruction.

```
[name]    IPW$RMS  {ADDNRM|GETBSC|DELBSC|DISALL|ADDALL|DELALL|
                    GETSNA|DELSNA|DELTMP}
                    [,msg-addr][,R5][,NMR=YES]
```

ADDNRM specifies to add the message pointed to by field 'TCMW' at the tail of the remote message queue of the remote id, specified in the low order byte of register 0. The message can be either in nodal message record format or in normal VSE/POWER format. In the last case, the first byte must contain the length of the message text.

GETSNA specifies to return the address of the 'head' message queued for the remote id, which is specified in the LUCB pointed to by register 8. Upon return, register 1 addresses the message or contains zero if no message is queued.

DELTMP specifies to remove the 'head' message from the message queue of the remote id, which is specified in the LUCB pointed to by register 8. The message is then added at the tail of the temporary 'delete' queue.

DELSNA specifies to delete all messages which are currently in the temporary delete queue of the remote id described by the LUCB pointed to by register 8.

ADDDEL specifies to add all messages currently in the temporary delete message queue of the remote id, addressed by the LUCB pointed to by register 8. The messages are added at the top of the remote message queue. Upon return, register 1 contains zero in case of an empty temporary delete queue. All message slots are freed.

DELBSC specifies to de-queue the message addressed by register 1 from the remote message queue of the remote id, contained in the LCB, addressed by register 9. The message slot is freed.

GETBSC specifies to return the address of the 'head' message from the message queue of the remote id, contained in the LCB, pointed to by register 9. Upon return, register 1 addresses the message or if no message is queued, register 1 is zero.

ADDALL specifies to add the message addressed by field 'TCMW' to the ALLUSERS message queue. The message can be either in nodal message record format or in normal VSE/POWER format. In the last case, the first byte must contain the length of the message text. The maximum message length is 59 bytes. Register 0 must contain the originating remote id in its high-order byte.

Upon return, register 1 contains zero if the ALLUSERS message queue is full.

DELALL specifies to delete either a specific or all ALLUSERS messages. Register must contain the binary remote id number in its low order byte and register 1 must contain the message number supposed to be deleted. If register 1 is zero, all messages are deleted.

If an attempt is made to delete a message but the requestor (remote id) is not entitled to do so, register 1 is set to zero, upon return.

DISALL specifies to return the address of first/next message in the ALLUSERS message queue. Register 1 must be zero for the first macro call. Upon return, register 1 contains the address of the first/next message. When all messages have been returned or the ALLUSERS message queue is empty, register 1 is set to zero.

msgaddr specifies the address of the message to be queued. The first byte of the message must contain the length of the message text (VSE/POWER format).

R5 specifies that register 5 contains the address of the TCB to be used for message modification. If omitted, the TCB of the requesting task is used for message modification.

NMR specifies whether the message to be written is in nodal message record (NMR) format or not.

YES specifies that the message is in NMR format.

NO specifies that the message is not in NMR format. This is the default if the parameter is omitted.

IPW$RQS - Reserve Queue Record

The IPW$RQS macro is used to allocate a queue record from the free queue record chain and a DBLK group from the free DBLK group chain.

Registers 14 and 15 are used as linkage registers.

```
[name]    IPW$RQS
```

IPW$RSR - Reserve Resource

The IPW$RSR macro is used to get exclusive use of a VSE/POWER resource, (for example DMB, MCB) and to lock the control block against concurrent use by other VSE/POWER tasks. If the macro call is unsuccessful then the task waits on the resource.

Register 2 is destroyed by execution of the macro instruction and register 3 contains the address of the control block concerned.

```
[name]    IPW$RSR  {symbol|(reg)}
```

symbol specifies the name of the control block to be reserved. This is the name of the field containing the address of the control block, as defined in the CAT, with the first two characters stripped off. If register notation is used, the designated register must have been previously loaded with the address of the control block concerned.

IPW$RSV - Reserve GETVIS Storage

The IPW$RSV macro is used to obtain the specified amount of storage from the GETVIS pool and to reserve this for use by the requesting task.

Registers 2 and 3 are destroyed by execution of the macro instruction.
Register 4 is destroyed when either ANCHOR or OWNER was specified.

Upon return, register 0 contains the return code as passed by GETVIS if WAIT=NO|COND was specified. In all other cases it will be zero. Register 1 contains the address of the requested user area or zero if no storage was available.

```
[name]    IPW$RSV  LENGTH=nnnnn|(reg)
                  [,POOL=GEN|MSG|NET|SNA|WACB|COCB]
                  [,WAIT=YES|NO|COND]
                  [,BDY=PAGE|NO]
                  [,OWNER=(reg)][,ANCHOR=(reg)]
                  [,PREFIX=YES|NO]
```

LENGTH nnnnn specifies the length in bytes of the storage. If register notation is used, the designated register must have been previously loaded with the length. 16 bytes will be added to the value specified to allow for the buffer control area. The length can range from 1 to (16M-16) bytes and is automatically rounded up to the next multiple of 128 bytes.

POOL specifies one of the VSE/POWER supported pool types. The types supported at the moment are MSG, GEN, SNA, NET, WACB, and COCB. The default is GEN.

WAIT specifies whether the task wants to wait if there is insufficient storage to satisfy the request or not.

YES is default and specifies that the task will wait until storage becomes available.

NO means that control will be returned to the task when no storage is available.

COND means that the task will wait until either storage becomes available or the task is forced to stop (termination code 'S' set in TCB).

Note: Only when no storage was available (the task does a wait IPW$WFC on the virtual storage control block) and the task is posted because storage has been freed by some task, is there a check for the 'S' code condition, upon which the task returns to the caller with register 1 = 0.

BDY PAGE specifies that the storage must be aligned on a page boundary. If not specified then the storage will be obtained in the first free space available for that POOL.

OWNER specifies the address of the TCB of the task that is to be used as owner for this storage. If not specified the issuing task will be considered as owner. Must NOT be specified if ANCHOR is specified.

ANCHOR specifies that the storage area is to be chained as the last entry in the queue whose head pointer is addressed by a doubleword pointed to by the designated register. Must NOT be specified if OWNER is specified.

PREFIX specifies whether or not if the storage area to be obtained is preceded by a VSE/POWER system prefix.

YES specifies to precede the storage area with a VSE/POWER system prefix, containing the length. This is the default when the parameter is omitted.

NO specifies that no prefix precedes the storage area.

Note: Even if WAIT=COND was specified, R1 must be checked to see if storage became available. It may happen that the task is posted again because of some abnormal condition, or a termination condition, and will then return without having acquired the storage.

IPW$RSW - Reserve Fixed (Real) Storage

The IPW$RSW macro is used to obtain an area of contiguous fixed storage. Upon return, register 0 contains:

- the real address, if desired,
- or zero if no storage available

and register 1 contains:

- the virtual address of the obtained storage area
- or the address of the storage control block ECB which is posted when storage is avaiable.

Registers 2 and 3 are destroyed by execution of the macro instruction.

```
[name]    IPW$RSW  size,[WAIT][,OWNER=SYS][,REALAD=YES|NO]
                       [,REQ=CUSH]
```

size specifies the number of bytes of storage to obtain. The parameter can be specified as any decimal digit up to the maximum allowed length. If register notation is used, the designated register must have been loaded with the length of the storage. If 'BF' is specified, storage is reserved in the length of the physical unit record buffer size (max. 2032 bytes).

WAIT specifies that the macro processing routine is to wait until storage becomes available. If the option is omitted and no storage is available, register 0 is set to zero and register 1 addresses an ECB which will be posted when storage becomes available. This may be used by the calling task if it is desired to wait until storage becomes available.

OWNER SYS specifies, that VSE/POWER is the owner of the acquired storage area rather than the task obtaining the storage. If the parameter is omitted, the task acquiring the storage, is registered as owner.

REALAD specifies either to return the real or virtual address of the storage in register 0.

YES specifies to return the real address of the obtained storage area. YES is the default if the parameter is omitted.

NO specifies to return the virtual address of the storage area.

REQ=CUSH specifies, that the storage area may be obtained from the real storage cushion. This parameter should be specified only for important functions that should work even in short on real storage state. For a list of currently identified important functions see module IPW$$I7 'SET UP REAL STORAGE CUSHION'.

IPW$SAV - Save Caller's Registers

The IPW$SAV macro saves registers 14, 15 and 0 through 12 in the save area addressed by register 13. No registers are destroyed by execution of the macro.

```
[name]    IPW$SAV
```

IPW$SRJ - Scan Reader JECL Statement

The IPW$SRJ macro is used to invoke the reader JECL processing routine in order to syntax check the JECL statement. The macro expansion establishes a linkage to the IPW$$SC routine.

Registers 0 and 1 must be setup as follows:

R0 address of column 72 of the JECL statement to be checked.
R1 address of the parameter to be checked.

Upon return, registers 0 and 1 are passed as follows:

R0 address of the parameter value, if valid, or negative address if the parameter is invalid.
R1 address of the next parameter delimiter.

Switch 'LWPI' is set on if the present parameter is also the last parameter.

Registers 14 and 15 are used as linkage registers.

```
[name]    IPW$SRJ
```

IPW$SRM - Set Remote Mask

The IPW$SRM macro is used to indicate in a shared spooling environment that a remote work station either logged on or logged off. Register 1 must contain the binary remote id number in its low order byte.

Registers 0 - 3 are destroyed by execution of the macro.

```
[name]    IPW$SRM   TYPE=LOGON|LOGOFF
```

LOGON specifies that the remote work station, named by register 1, logged on.

LOGOFF specifies that the remote work station, named by register 1, logged off.

IPW$SSJ - Call Parameter Checking Routine

The IPW$SSJ macro provides the linkage to the parameter checking routine (IPW$$PC).

Registers 14 and 15 are used as linkage registers.

```
[name]    IPW$SSJ  {DSECT|CARRIER=subtype}

                     or

[name]    IPW$SSJ  PWR=(field,[subchar],para,format,[flag],[fb],
                        [flgpara])
```

CARRIER generates a call to the IPW$$PC module. The 'subtype' defines the carrrier of the parameter(s) to be checked. 'subtype' is the name of the appropriate equate defined in the parameter list DSECT. Register 1 must address the PC parameter list. Register 0 is used to contain the carrier type. If the CARRIER is omitted, the macro assumes that the carrier type is already contained in register 0. Registers 14 and 15 are used as linkage registers.

DSECT causes the generation of the parameter list DSECT used as interface for the parameter checking routine.

PWR specifies to generate an entry of the syntax checking driver table of the spool parameter list (PWRSPL).

field specifies the PWRSPL field label where the parameter is to be found.

subchar specifies the substitution character, if any

para specifies the parameter definition table entry label

format specifies the parameter format, where

E means EBCDIC characters
T means EBCDIC text (imbedded blanks are allowed)
B means binary
F means flag bit
O means other

flag specifies the flag bit equate, if FORMAT=F.

fb specifies the feedback code to be returned when the parameter is wrong.

flgpara indicates that the presence of the parameter is controlled by a PWRSPL flag.

IPW$STM - Set Timer Interval

The IPW$STM macro is used to setup a timer interval the task wants to wait on or to cancel a previously set up timer interval.

Upon return, registers 0 - 3 are destroyed.

```
[name]    IPW$STM  TIME=ttt|(reg)[,WAIT=YES|NO]
                  [,TQE=(reg)][,ECB=(reg)]

                          or

[name]    IPW$STM  CANCEL=YES,TQE=(reg)
```

ECB specifies the address of a 4-bytes field, used as ECB, which is posted when the time interval expires. Registers 1 - 3 cannot be used when the TQE parameter is omitted. R1 cannot be used when TQE parameter is specified.

TIME ttt is the time interval in tenths of a second. If register notation is used, the designated register must have been previously loaded with the time interval. R1 cannot be used at all.

TQE specifies the address of a previously acquired TQE. If the parameter is omitted, storage for the TQE is reserved and register 1 is used as pointer register. The TQE must be in real storage.

WAIT specifies whether the requesting task wants to wait or not.

YES means that the macro expands into a wait and the TQE storage is automatically released. YES is default.
NO means that no wait is automatically generated.

CANCEL YES specifies to delete a time interval which has been already setup. TQE= must also be specified.

IPW$SXJ - Scan Execution JECL Statement

The IPW$SXJ macro is used to invoke the execution JECL processing routine in order to syntax check the JECL statement. The macro expansion establishes a linkage to the IPW$$XJ routine.

The address of the JECL statement to be processed must have been previously stored in 'TCRV' of the TCB and its length in 'TCRL'.

Upon return, register 15 contains the return code:

0 ok, JECL statement processed.
4 error occurred; no valid JECL statement.

Registers 14 and 15 are used as linkage registers.

```
[name]    IPW$SXJ
```

IPW$TDM - Switch Turbo Dispatcher Mode

The IPW$TDM macro is used to let processing of the VSE/POWER Maintask continue either as a parallel (PU) work unit or as a non-parallel (NP) work unit - in other words, to request switching to parallel or non-parallel processing mode.

Register 0, 1, and 2 are destroyed by execution of this macro instruction. The acquired processing mode is recorded by the tasks's TCF16NP flag.

Mode switching is ignored by the called service, when

- the Turbo Dispatcher is not activated (Standard Dispatcher instead)
- VSE/POWER has not been started with the SET WORKUNIT=PA autostart option, that means when VSE/POWER operates NP exclusively (default)

Mode switching is not actually requested, when the desired processing mode is already active.

```
[name]    IPW$TDM  {NP|PU}
```

NP specifies to enter a non-parallel work unit for the calling VSE/POWER subtask

PU specifies to enter a parallel work unit for the calling VSE/POWER subtask

IPW$TTM - TD-Subtask Timer Interval Support

This access macro is to support the TD-subtask with timer interval support.

```
[name]    IPW$TTM  STXIT=YES

[name]    IPW$TTM  TIME=(Rx),TQE=address

[name]    IPW$TTM  CANCEL=YES,TQE=address

[name]    IPW$TTM  PROCESS=YES

[name]    IPW$TTM  WAIT=(Rx)
[name]    IPW$TTM  WAIT=(Rx,REACTIVATE)
```

STXIT initializes the VSE Timer STXIT interface for the SETIME macro used for the other support macros.

TIME allows the caller to indicate a timer interval in tenths of a second following which an ECB is posted in the indicated TQE element and the Driver Subtask is also posted. The timer interval is contained in the register Rx. For the first request or any following request whose interrupt is to occur sooner in time then the previous open requests, a SETIME macro is issued to cause the internal TSIM routine to be executed which posts the Driver Subtask.

CANCEL the caller indicates that a previous IPW$TTM TIME= request is to be cancelled.

TQE specifies the address of a previously acquired TQE.

PROCESS called by the Driver Subtask following posting. It searches for any expired requests and, if any, reissues the SETIME macro for the soonest of any remaining requests.

WAIT allows the Driver Subtask to indicate it wishes to go into a wait state until it is posted by either the expiration of a SETIME interval request for the WAIT= interval (in tenths of a second), or by any other event which may occur sooner, with the register Rx containing the interval value.

REACTIVATE allows the Driver Subtask to indicate it wishes to go into a wait state as for the IPW$TTM WAIT=(Rx) macro, and additionally the macros IPW$TTM STXIT=YES and IPW$TTM PROCESS=YES are called immediately following.

IPW$TTS - SD-Subtask Timer Interval Support

This access macro is to support the SD-subtask with timer interval support.

```
[name]    IPW$TTS  STXIT=YES

[name]    IPW$TTS  TIME=(Rx),TQE=address

[name]    IPW$TTS  CANCEL=YES,TQE=address

[name]    IPW$TTS  PROCESS=YES

[name]    IPW$TTS  WAIT=(Rx)
[name]    IPW$TTS  WAIT=(Rx,REACTIVATE)
```

STXIT initializes the VSE Timer STXIT interface for the SETIME macro used for the other support macros.

TIME allows the caller to indicate a timer interval in tenths of a second following which an ECB is posted in the indicated TQE element and the Driver Subtask is also posted. The timer interval is contained in the register Rx. For the first request or any following request whose interrupt is to occur sooner in time then the previous open requests, a SETIME macro is issued to cause the internal TSIM routine to be executed which posts the Driver Subtask.

CANCEL the caller indicates that a previous IPW$TTS TIME= request is to be cancelled.

TQE specifies the address of a previously acquired TQE.

PROCESS called by the Driver Subtask following posting. It searches for any expired requests and, if any, reissues the SETIME macro for the soonest of any remaining requests.

WAIT allows the Driver Subtask to indicate it wishes to go into a wait state until it is posted by either the expiration of a SETIME interval request for the WAIT= interval (in tenths of a second), or by any other event which may occur sooner, with the register Rx containing the interval value.

REACTIVATE allows the Driver Subtask to indicate it wishes to go into a wait state as for the IPW$TTS WAIT=(Rx) macro, and additionally the macros IPW$TTS STXIT=YES and IPW$TTS PROCESS=YES are called immediately following.

IPW$ULP - Update LUB/PUB Tables

The IPW$ULP macro is used to invoke the LUB/PUB update routine to perform one of the following:

- To release the assignment for a given logical unit and to release ownership.
- To locate the PUB entry for a physical device to establish ownership.
- To locate a free LUB entry and to assign it to a given physical unit.
- To release all logical assignments to a given physical device.
- To release ownership of a physical device.
- To identify the physical device corresponding to a given logical unit (SYSxxx).
- To assign SYSLST to a given physical device.
- To unassign SYSLST from a given physical device.
- To assign a free LUB to a given physical device.
- To inform the supervisor about devices being spooled by VSE/POWER.
- To inform the supervisor about devices which are no longer spooled by VSE/POWER.

Registers 0 - 3 are used as parameter registers.

Registers 14 and 15 are used as linkage registers.

```
[name]    IPW$ULP
```

IPW$UNV - Unchain Virtual Storage Element

The IPW$UNV macro is used to unchain a specific element of a specified queue of virtual storage elements and to chain it to another queue. The address of the data part of the unchained element is returned in register 1. The element may be directly addressed or if no address is given then the first element of the queue is unchained.

Upon return, registers 0 - 3 are destroyed. If TO= was specified then
register 4 is also destroyed.

If register 1 is zero after the return, then unchain has failed.

```
[name]    IPW$UNV [ADDR=(reg)][,FROM=(reg)][,TO=(reg)][,LOCK=NO]
```

ADDR specifies the address of the data part of the element to be unchained. This parameter is optional and if not specified, or the register contains zero, then the first element of the queue specified by the FROM field is unchained and its address is returned in register 1. If register 1 is zero, no element could be found, i.e. neither the specified element nor any other element could be found in the chain.

FROM specifies the address of the head/tail pointer of the queue (TCHD) from which the element must be unchained. This parameter is only required when the first element is to be unchained, i.e. ADDR is NOT specified. If not specified, the virtual storage chain of the issuing task is considered as the FROM queue.

TO specifies the address of the head/tail pointer of the queue (TCHD) to which this unchained element must be chained at the end of the queue. If not specified, the issuing task with its virtual storage chain is considered as the TO queue.

LOCK Use of this operand means that no implicit locking of the virtual storage control block (VSCB) will be done. Specify LOCK=NO only if concurrent access to the relevant queues is controlled by reservation of another resource before the subject macro is used; the task linkage register save area is used to store registers 0 to 9, and 14 to 15!

Note: Head and tail pointers addressed by the 'FROM' or 'TO' operands both point to the virtual storage header part of the first or last element of a queue.

IPW$VCA - Validate Command Authorization

The IPW$VCA macro is used to examine if the command issuer is authorized to execute the command or not.

Return is made with a displacement of zero when the issuer is not authorized and with a displacement of four if the command issuer has enough authority.

Registers 14 and 15 are used as linkage registers; register 0 is destroyed by execution of the macro instruction.

Note: The macro is only applicable for the VSE/POWER command processor.

```
[name]    IPW$VCA  command[,type]
```

command specifies the command to be processed (e. g. PSTART or PSTOP).

type specifies the type of command to be processed. The following types are supported: PART, JOB, RJE, DEV, XTASK, INT, TASK, PNET, MSG, A, Q, M, CUU or VIO.

IPW$VDA - Validate Data Area Address

The IPW$VDA macro is used to validate if the CCW and associated data area lie in the user's partition or any other area the user is allowed to access (LTA, SVA, dynamic partition GETVIS area). In addition, the CCB is validated not to specify the usage of Format 1 CCW.

Register 8 must point to the CCB in question, register 6 must address the partition control block of the partition concerned and register 4 must address the spool entry with the partition control block. If running with an ESA supervisor, the access-register mode must be set on.

Registers 1 - 3 are destroyed by execution of the macro. Upon return, register 0 contains one of the following codes:

00 validation ok
04 error occurred: CCW or data area outside of allowed area, or Format 1 CCW specified. TCERC contains a more specific error code.

```
[name]    IPW$VDA  [VD=YES|NO]
```

VD specifies whether to validate the CCW only or also the associated data area.

YES causes to validate both CCW and data area.

NO causes to validate the CCW only.

IPW$WF[x] - Wait for VSE/POWER Event

The IPW$WF[x] macros are used to place the task in a VSE/POWER wait condition. 'x' specifies the event for which the task is waiting.

```
[name]    IPW$WF[x]  [ecbname|(r1)]
```

x specifies the event for which the task is to wait as one of the following:

C specifies that the task is waiting for posting of the traffic bit (bit X'80' of ECB byte 2) of the ECB addressed by register 1. The posting must eventually occur. This service should be used to wait for I/O completion, because bit X'20' or ECB byte 2 is also checked for unrecoverable I/O error whereupon task specific action is taken.

S specifies that the task is waiting for posting of the traffic bit (bit X'80' of ECB byte 2) of the ECB addressed by register 1. The posting need not necessarily occur at all. This service should be used to wait for I/O completion, because bit X'20' or ECB byte 2 is also checked for unrecoverable I/O error whereupon task specific action is taken.

E specifies that the task is waiting for posting of the traffic bit (bit X'80' of ECB byte 2) of the ECB addressed by register 1. The posting must eventually occur.

```
[name]    IPW$WF[y]  [ecblistname|(r1)]
```

y specifies the event for which the task is to wait as one of the following:

M specifies that the task is doing a multiple wait for posting of the traffic bit (bit 16) of any of a set of control blocks. The addresses of the relevant control blocks are contained in a sequential list addressed by register 1 and delimited by X'FF'. The posting must occur.

Q specifies that the task is doing a multiple wait for posting of the traffic bit (bit 32) of any of a set of control blocks. The addresses of the relevant control blocks are contained in a sequential list addressed by register 1 and delimited by X'FF'.

Note: The relevant control blocks are typically class anchors of the VSE/POWER RDR/LST/PUN/XMT queues, therefore IPW$WFQ stands for 'queue' posting. 'Q' state implies that none of the conditions need occur.

X specifies that the task (typically the dynamic partition scheduling task) is doing a multiple wait for posting of the ECB traffic bit (bit 16) of a simple (first) control block and/or for posting of the traffic bit (bit 32) of any of a set of control blocks. The addresses of the relevant control blocks are contained in a sequential list addressed by register 1 and delimited by x'FF'.

```
[name]    IPW$WF[z]
```

z specifies the event for which the task is to wait as one of the following:

B specifies that the task is waiting for posting of RJE,BSC or PNET event.

D specifies that the task is waiting for immediate dispatch. The macro instruction is used to enter the VSE/POWER dispatcher in order to give other higher-priority tasks the chance to get control.

I specifies that the task is inactive and waiting for initiation.

L specifies that the task is waiting for a resource which is presently locked against concurrent use. Register 3 addresses the resource the task is waiting on.

O specifies that the task is operator bound and waits for a PGO/PSETUP command.

IPW$WQR - Write Queue Record

The IPW$WQR macro is used to update the queue record, addressed by the I/O request word, in the storage copy of the queue file and to write the queue record block, containing the updated queue record, back to disk. Register 1 must point to the I/O request word. This macro must be issued when the status or attributes of a particular queue record have been changed rather than just some chaining pointers.

Registers 0 - 3 are destroyed by the execution of the macro instruction.

```
 [name]   IPW$WQR   {address|(reg)}
```

address specifies the address of the I/O request word containing the relative queue record number and the address of the storage area which holds the queue record to be written back to disk.

(reg) specifies that the address of the I/O request word is contained in the designated register.

Note: Length in "I/O request word" need not be set, Q-F-server always operates at length of Q-record.

IPW$WTD - Write Data Block to Disk

The IPW$WTD macro is used to write the data block, addressed by the I/O request word, to the VSE/POWER data file.

Registers 0 - 3 are destroyed by execution of the macro.

```
 [name]   IPW$WTD   {address|(R1)}[,WAIT=FORCE]
```

address specifies the address of the I/O request word used for the write operation.

(R1) specifies that the address of the I/O request word (12 bytes) is contained in register 1.

WAIT FORCE specifies that the disk service routine must wait for the I/O completion regardless of any 'double buffering' flag set.

Note: The I/O request word provided length field will be used by disk service as specified, to copy the virtual DBLK area to the real I/O area - and pad it with x'00', if specified length is smaller than the DBLK size.

IPW$WTO - Write to Operator

The IPW$WTO macro is used to write a message to the system operator, or to a spool-access user, if the task uses the spool-access support.

To write a message to the system operator even if a task uses the spool-access support, TCPCOP should be set on in TCMW.

The message should contain only uppercase characters. If a user written exit routine writes a non-VSE/POWER message:

- TCDNMM must be set on in TCMW in order to avoid the modification of the message by VSE/POWER (and then afterwards reset),
- TCPCOP must be set on in TCMW in order to insure that the message is routed to the central operator if desired (and then afterwards reset), and
- the message routing and descriptor codes may optionally be set in the TCB (fields TCMRT and TCMDC, automatically reset).

Registers 0 - 3 are destroyed by execution of the macro.

```
[name]    IPW$WTO  {TCMW|msgaddr|RC=YES}[,HOLD][,R5][,NMR=YES|NO]
```

msgaddr specifies the address of the message to be issued. The first byte of the message must contain the length of the message text (VSE/POWER format). The message text must not exceed 120 bytes.

TCMW specifies that the address of the message to be issued is present in the message control word of the TCB (TCMW).

HOLD specifies to keep the lock for the local/remote message control block. The appropriate message control block must be explicitly released when no longer needed.

R5 specifies that register 5 contains the address of the TCB to be used for message modification. If omitted, the TCB of the requesting task is used for message modification.

NMR specifies whether the message to be written is in nodal message record (NMR) format or not.

YES specifies that the message is in NMR format.

NO specifies that the message is not in NMR format. This is the default if the parameter is omitted.

RC YES specifies that the address of the message (VSE/POWER format) is contained in register 1 and the return code to be inserted into the message is in register 0.

IPW$WTQ - Write Queue Record Block to Disk

The IPW$WTQ macro is used to write the queue record block, addressed by the I/O request word, to the VSE/POWER queue file.

Registers 0 - 3 are destroyed by execution of the macro.

```
[name]   IPW$WTQ   {address|(R1)} [,LOCK=YES|NO][,ERROR=IGN]
```

address specifies the address of the I/O request word used for the write operation.

(R1) specifies that the address of the I/O request word (12 bytes) is contained in register 1.

LOCK YES specifies that the disk service routine, contained in the VSE/POWER nucleus performs a MCB release operation after processing the requested function; this is the default.

NO means that the calling task already owns the MCB and that the MCB will not be released at completion of the I/O.

ERROR IGN causes the disk service to return to the caller after an I/O error occurred. The I/O error will then not be handled by the I/O error handler. This is only applicable when writing the master record back to disk. It also causes to attempt to write the master record even so the 'queue file damaged' flag is set in the master record.

Note: The I/O request word provided length field will be used, if the Q-record-block number is the block number of the master record; if not, then disk service will operate with the constant "queue record block length".

IPW$WTR - Write to Operator with Reply

The IPW$WTR macro is used to write a message to the system operator and to wait for his reply.

Registers 0 - 3 are destroyed by execution of the macro.

```
[name]    IPW$WTR  TCMW|{msgaddr,repaddr}[,R5]
```

TCMW specifies that the address of the message to be issued as well as the address of the reply area are present in the message control word of the TCB (TCMW and TCAW). The first byte of the reply area must contain the length of the reply area.

msgaddr specifies the address of the message to be issued. The first byte of the message must contain the length of the message text. The message text must not exceed 120 bytes.

repaddr specifies the address of the reply area to be used for the reply from the operator. The first byte must contain the length of the reply area. The operator's reply is automatically translated to uppercase characters.

R5 specifies that register 5 contains the address of the TCB to be used for message modification. If omitted, the TCB of the requesting task is used for message modification.

IPW$WTT - Write Tape Record

The IPW$WTT macro is used to write a record to tape, described by the tape control block (TBB), which is pointed to by the TCB of the requesting task.

Registers 0 - 2 are destroyed by execution of the macro instruction.

```
[name]    IPW$WTT  {TCQW[,PREL]|TCDW|(R1)}
```

PREL specifies to write a queue record to tape using the length of a Previous RELease

TCQW specifies to write the queue record, addressed by the TCB (field TCQW), onto tape with the following length:

1. 'current systems queue-record length',if PREL not specified, or if PREL specified and TCB field TCOQRL contains zero.
2. 'length as specified in TCOQRL', if PREL is specified and TCB field TCOQRL contains a non-zero value which typically specifies the queue record length of a previous VSE/POWER release - for details refer to the POFFLOAD BACKUPnn/SAVEnn command.

TCDW specifies to write the data block (DBLK), addressed by the TCB (field TCDW) onto tape.

(R1) specifies to write the record, addressed by register 1 onto tape. The length of the record must have been stored in the TBB before execution of the macro instruction.

Appendix D. VSE/POWER Storage Requirements for Release 6.1

The following was formally located in the *VSE/POWER Administration and Guide* manual.

Note: The following table is correct only for the Version 6.1 of VSE/POWER.

Figure 158. Values for Calculating IBM VSE/POWER's Fixable and Getvis Areas

VSE/POWER Component or Task	Fixable Area (bytes)	Getvis Area (bytes)
Dynamic control blocks (always required)	26,000 (see Note)	
Every further data file extent	D	
PNET control blocks for networking	192	S+V+Z
SNA control block for RJE,SNA support	192	
Every local writer task		
With one buffer	928+P+A	512+N
With two buffers	928+2P+A	512+N
With four buffers	928+2P+A	512+2N
Every local punch task	928+P+A	256+N
Every local card reader task		
With one input buffer	680+P	256+F+N
With two input buffers	680+2P	256+F+N
Diskette I/O unit connected	928+P+E	256+F+N
Every local 3540 reader task	928+E	256+F+N
Every print status task attached		
. by temporary command processor	550	960
. by permanent command processor	1500	
Every temporary command processor task	608	2,240
Every local tape reader task	608	4,352+N
Every off-load task		
Save (backup) function	544	256+N
Load function	544	12,544
Select function	544	12,800
Cross-partition support		
Reader task (PUTSPOOL)	740	256+F+N
Writer task (GETSPOOL)	740	256+F+N
Control task (CTLSPOOL)	640	
Every device-service task	680	1110+N+J
Every spool-access-support connection		
CTL function	640	780+J
GET function	704	1,024+N+J
PUT function	704	1,024+N+J
GCM function	640	780+J
RJE,SNA tasks		
Every reader task	640	256+F+N
Every writer task	640	256+N
RJE,BSC tasks		
Every reader task	640	256+F+N
Every writer task	640	256+N
Every execution reader (one per partition)	604	832+N+X+Y
Every execution writer (up to 28 per partition)	604+C	660+N
First RJE,BSC line	352+1,856	
Every additional RJE,BSC line	1,856	
Every job/output transmitter	544	1,536+N
Every console transmitter	544	1,280
Every job/output receiver	544	2,560+N
Every session (PNET,BSC)	1,152+G	2,176
Every session (PNET,SNA)	960	3,200+H
Task trace (if ever required), default	2,048	

Note: The above numbers may in reality vary slightly from those shown.

Note: A certain part of this storage depends on the number of queue records. The value given here assumes a queue file of 1000 records. Every additional 1000 records (up to 32,767) require an additional 130 bytes. Likewise, this storage is calculated under the assumption of one data file extent with a DBLK size of 4080. Furthermore, this storage contains a real storage cushion of 4,5KB.

Legend for Figure 158 on page 796 :

```
A = 96 if printing/punching from spool tape; otherwise A = 0

C = 96 if spooling to tape; otherwise C = 0

                      DBLK + 39
D = 32 x n  where n = ---------  (rounded to the next higher integer)
                         32

                      (R+8) x 26
E = 32 x e  where e = ----------  (rounded to the next higher integer)
                          32

                  R = logical record length as specified for the
                      diskette I/O unit

F = 256 (needed only for building control records)

G = ((1 + tr x bt) + (1 + rv x br)) x bs

H = ((1 + tr x bt) + (1 + rv x br)) x bs
    where for G and H:
    tr = Number of active job/output transmitters
    bt = Number of buffers per active job/output transmitter
    rv = Number of active job/output receivers
    br = Number of buffers per active job/output receiver
    bs = For G: Buffer size + 28 (rounded up to next multiple of 32)
    bs = For H: Buffer size + 128 (rounded up to next multiple of 128)

J = Size of the spool-access support buffer as defined for a specific
    function request (up to a maximum of 65,536 bytes)

N = DBLK + 16 rounded up to the next multiple of 128 bytes
    where DBLK = The value specified in DBLK=n of the POWER generation
    macro

                      DBLK + 39
P = 32 x n  where n = ---------  (rounded to the next higher integer,
                          32      but not to exceed 4096)

S = 256 if shared spooling is specified, otherwise S = 0
V = 256 if any connection is started using SDLC line discipline;
    otherwise V = 0

X = 3200 if SLI statements are to be processed; otherwise X = 0

Y = 128 for every additional SLI nesting level; otherwise Y = 0

Z = n x 40 + 116 (rounded up to the next multiple of 128)
    where n = Number of nodes defined in the network definition table
```

List of Abbreviations

ACC	Account control block
ACB	Access method control block
AQRA	Auxiliary queue record area
ASAB	Asynchronous service anchor block
BCA	Buffer control area
BCW	Buffer control word
CAT	Control address table, or Permanent area.
CB	Control block
CCB	Command control block
CI	Control interval for FBA
CIB	Communicator information block
CIE	Communicator information element
CIDF	CI description field
COCB	Compaction table control block
CP	Command processor
CPCB	Command processor control block
DLRSA	Double linkage register save area
DMB	Disk management block
DSHR	Data set header record
DRW	Disk request word also called I/O request word
DPCB	Dynamic partition control block
DPST	Dynamic partition scheduling task
EAR	Execution account record
EDCB	External device control block
ECB	Event control block
ETX	End of text
FBA	Fixed block architecture
FCB	Forms control buffer
FCS	Function control byte
GNB	Generation table
INIT/TERM	Initiator/Terminator
JAI	Job Accounting Interface
JECL	Job entry control language
JHR	Job header record
JTR	Job trailer record
LCB	Line control block
LDA	Logical data record area
LK	Lockword
LL	Logical list
LMF	Line manager field
LMGR	Line manager
LRCB	Logon request control block
LRSA	Linkage register save area
LST	List
LTA	Logical transient area in the VSE supervisor
LUCB	Logical unit control block
LW	Logical writer
MCB	Module control block
MCTA	Master class table area
MECB	Master (main) event control block
MEDCB	Master external device control block
MLI	Multi-leaving
MLT	Master line table
MMB	Message control block
MR	Master record
MRA	Master record area
MSCB	Remote message control block
NAT	Node attached table
NCB	Node control block
NDT	Network definition table
NMR	Nodal message record
NPGR	Negative permission granted record
NQ	Get next from queue
OPDE	Output parameter definition entry
OPTB	Output parameter text block
PDB	Partition control block
PDA	Physical data record area
PGR	Permission granted record
PNCB	PNET master control block
PSA	Partition save area
PUN	Punch
PWS	Physical work space
QRA	Queue record area
RCF	Record control field
RDF	Record description field for FBA device
RDR	Reader
RE	User reader exit routine
RIF	Request to initiate a function
RJE	Remote job entry
RMCB	SNA remote control block
RPL	Request parameter list
RTAM	Remote terminal access method
SAM	Sequential access method
SAS	Spool Access Support
SCB	Storage control block or string control byte
SDA	Single data adapter
SDCB	PNET SSL Driver Control Block
SEH	Spool environment header
SER	Spool environment record
SKAD	Seek address
SLA	Separator line area
SLW	SLI work space
SNCB	SNA control block
SPB	Spool environment block
SPL	Spool parameter list
SPM	Spool management
SRB	Service request block
SRCB	String record control byte
SUCB	SNA unit control block
TBB	Tape control block
TCB	Task control block
TDCB	PNET TCP Driver Control Block
TMF	Task management field
TMS	Task management service
TR	Task terminator
TRSA	Task register save area
TSL	Task selection list

VSCB	Virtual storage control block
VTAM	Virtual telecommunications access method
WACB	SNA work space
WCB	Wait control block
WTR	Writer

Bibliography

To use this manual effectively, you should be familiar with the concepts and facilities of VSE/AF described in the following manuals:

VSE/ESA:

- *VSE/ESA Planning*, SC33-6703
- *VSE/ESA Installation*, SC33-6704
- *VSE/ESA Guide to System Functions*, SC33-6711
- *VSE/ESA Operation*, SC33-6706
- *VSE/ESA System Control Statements*, SC33-6713
- *VSE/ESA Diagnosis Tools*, SC33-6614
- *VSE/ESA Quick Reference*, GX33-9026

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

- *Planning for NetView, NCP, and VTAM*, SC31-7122
- *VTAM Programming*, SC31-6496

Other VSE/POWER publications are:

- *VSE/POWER Administration and Operation*, SC33-6733
- *VSE/POWER Application Programming*, SC33-6736
- *VSE/POWER Remote Job Entry*, SC33-6734
- *VSE/POWER Networking*, SC33-6735

PNET users should also be familiar with the NJE protocols as described in:

- *Network Job Entry Formats and Protocols.*, SC23-0070

Glossary

Following is a definition of some of the terminology used in this manual.

Adjacent Node. Adjacent nodes are any nodes which are directly connected with one another by a BSC connection or an SDLC session.

Alternate Route. It is possible to define for any destination an alternate path that may be used in the case that the main path, defined by the ROUTE1 parameter in the PNODE macro, is not available. This second route is called an alternate route. It may be specified by the ROUTE2 parameter in the PNODE generation when defining the destination node. It does not need to be a path using the same line discipline. It is NOT used as a 'load levelling' mechanism.

Command Switching. Command switching is the transmission of a command which has been received from the network to the next destination on its way to its final destination.

Compression. Compressing a data stream replaces two or more consecutive blanks by a one byte control character containing information as to how many blanks have been compressed. If three or more consecutive like non-blank characters are found then they are replaced by two bytes, the first control byte containing information as to how many occurrences of the character have been compressed, and the second byte containing the actual character.

Compression is very widely used to reduce the size of the data stream before transmitting it via a teleprocessing system.

Decompression. Decompression takes a compressed data string and expands it again to its original size by replacing the compression control bytes by the specified number of characters or blanks.

Direct Link. A direct link is defined as a connection between two adjacent nodes which is physically accomplished by a BSC line between the two nodes.

End Node. The end node refers to that node which is designated as the final destination for the job or output. This can be the local node or any other node that is reachable within the network.

Execution Node. The execution node is the end node for jobs. It is that node on which the job will be executed. It may be another VSE/POWER node or any other node supporting the networking protocol used by VSE/POWER PNET, and reachable from the local node.

Final Destination. Same as end node.

Intermediate Node. An intermediate node is any node which is not an end node. It is used as a temporary store for jobs or output which are destined for another node to which there was no direct path established from the originating node.

Local Node. The local node is that node of the network at which the user (or reader of the specifications) is assumed to be sitting. It is that name defined in the PNODE macro with the LOCAL=YES parameter specified.

Message Switching. Message switching is the transmission of a message which has been received from the network to the next destination on its way to its final destination. If there is no link active to the next required node then the message is thrown away.

Network Control Records. A network control record in PNET is one of the following record types:

- Job header record
- Data set header record
- Job trailer record

For a description of the various record types, please refer to the index for the appropriate record.

Nodeid. A nodeid is a 1-8 byte alphanumeric identifier, the first character of which must be alphabetic, which is used to identify the node within the network.

Originating Node. The originating node is that node within the network where a job was entered into the system, or where output was produced. It may be the same as the end node, in which case it means that the job or output is never transmitted via the network.

Primary Route. This is the route which will always be taken to reach a final destination if the link or session is active and signed-on. It is specified by the ROUTE1 parameter in the PNODE macro.

Segmentation. A record or data set is segmented if it is split into two or more parts. Records are often segmented because they exceed an allowed maximum value. Each segment is preceded by control information specifying the length of the segment and perhaps whether this is first, middle, or last segment.

A data set may be segmented to reduce the amount of working storage required to hold that data set or in the case of VSE/POWER output segmentation, to reduce the size of the output data set being produced and to allow this to be printed or punched before the complete job is finished.

Spanned Records. Spanned data records may occur when a logical data record is larger than the physical data record, or when the logical record is too large for the remaining space in a buffer. In this case it is possible to say that the records will be 'spanned' between blocks. That means that a part of the logical record will be put into the first block and the remainder of it will be placed at the start of the next block. Spanning may take place over any number of physical data records. To recover the logical record more than one physical record will have to be read.

Session. A session can exist between any two nodes in the network. A session is established by use of VTAM.

Store-and-Forward Node. Same as an intermediate node.

Topology Record. Topology records are written by JES2 NJE to dynamically describe the network. Records are sent whenever a node is started or stopped. These records are ignored by VSE/POWER.

Userid. The userid is a 1-8 byte alphanumeric identifier which may be used to identify the user who has submitted, or is to receive, the job or output.

Index

File Number: S/370 9370-37
Program Number: 5686-CF7

Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.

SC33-6322-07