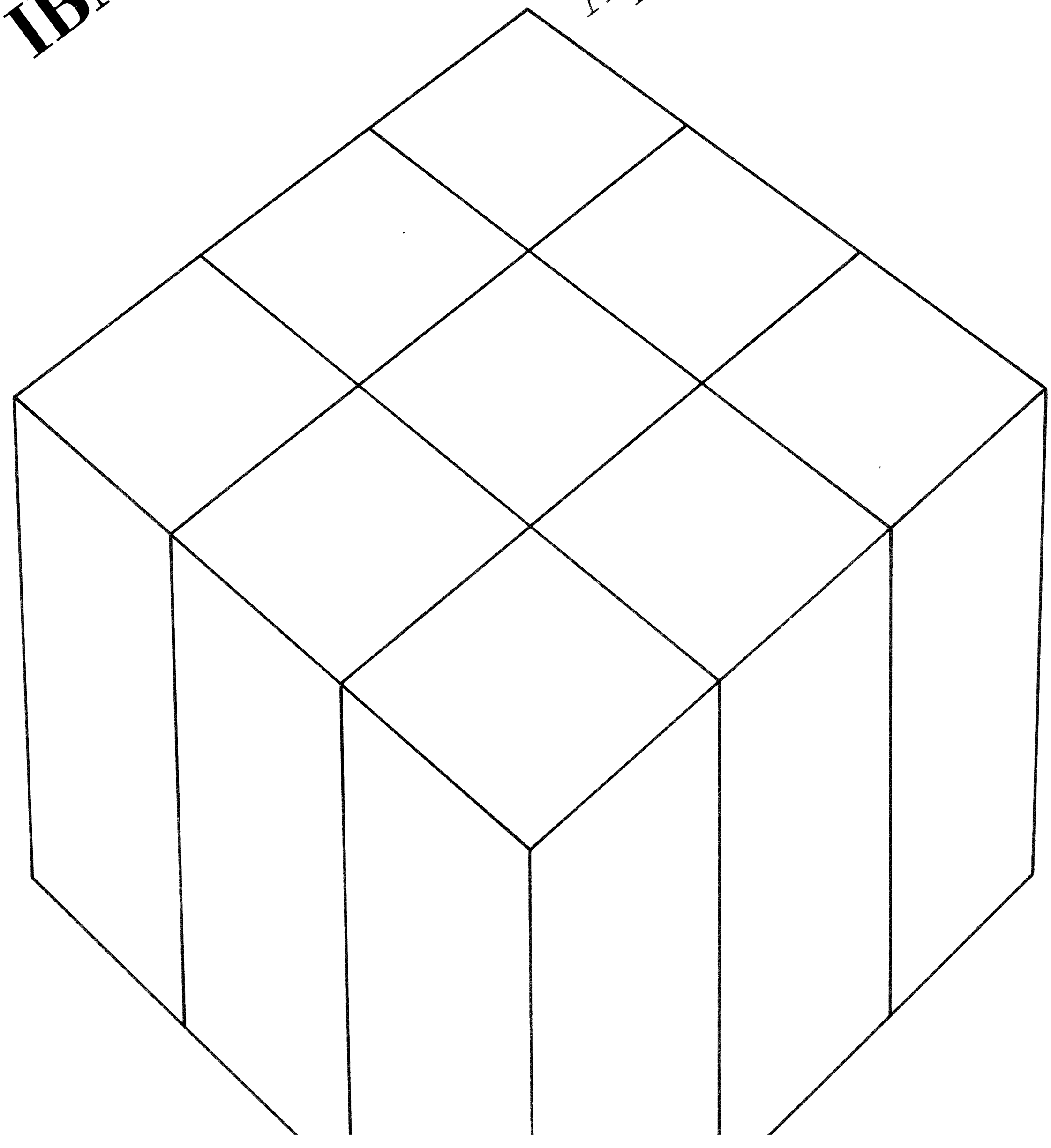
IBM Virtual Storage Extended
POWER

Application Programming



IBM Virtual Storage Extended/ POWER

Application Programming

Version 2 Release 3

Program Number 5666-273

Order Number SC33-6276-00
File No. S370/4300-40

First Edition (June 1987)

This edition is an excerpt from the VSE/POWER Installation and Operations Guide, SH12-5329-5. Changes or additions to the text and illustrations against that manual are indicated by a vertical line to the left of the change. This edition applies to Version 2, Release 3 of IBM Virtual Storage Extended/POWER, Program Number 5666-273, and to all subsequent releases until otherwise indicated in new editions or Technical Newsletters.

Changes are made periodically to the information herein; before using this publication in connection with the operation of IBM systems, consult the latest IBM System/370, 30XX and 4300 Processors Bibliography, GC20-0001, for the editions that are applicable and current.

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 licensed program in this document is not intended to state or imply that only IBM's licensed program may be used. Any functionally equivalent program may be used instead.

Publications are not stocked at the addresses given below; requests for copies of IBM publications should be made to your IBM representative or to the IBM branch office serving your locality.

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

IBM Corporation
Dept. 6R1BP
180 Kost Road
Mechanicsburg, PA 17055, USA

or to:

IBM Deutschland GmbH
Dept. 3164
Schoenaicher Strasse 220
D-7030 Boeblingen, Federal Republic of Germany

IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

PREFACE

This manual describes application programming functions of IBM Virtual Storage Extended/POWER (VSE/POWER for short), a program that schedules the input and output of jobs. The manual provides guidance and reference information for the major user task application programming.

The other user tasks, such as planning, installation, operation, and diagnosis are described in the companion volume VSE/POWER Installation and Operations Guide, from which this manual has been extracted.

This book contains information about:

- Controlling job accounting and output segmentation from within your application programs.

- The use of VSE/POWER services in application programs running in one of the system's partitions, under or outside the control of VSE/POWER.

- Writing exit routines.

This book is organized in chapters and appendixes which have been taken over from the VSE/POWER Installation and Operations Guide, SH12-5329-5:

Chapter 1, part of the former Chapter 5 of SH12-5329-5, describes how to do job accounting and output segmentation.

Chapter 2, the former Chapter 7 of SH12-5329-5, describes the support available for accessing VSE/POWER services from an application program running in one of the system's partitions.

Chapter 3, the former Chapter 8, provides the information needed to have an application program write spooled output to a device under control of that program.

Appendix A provides information on coding and using reader exit routines.

Appendix B describes the XECB macro based support for cross-partition communication. This support is still available to ensure program compatibility.

At the end of this book, you find:

- A **Glossary** which explains terms used in this manual. For terms not explained in this manual, you may refer to the IBM manual Vocabulary for Data Processing, Telecommunications, and Office Systems, GC20-1699.

- A **Bibliography** which lists related documentation.
- An **Index** to aid you in retrieving information from this manual.

Throughout this book:

> Version and release numbers for VSE/POWER and various other products are shown in the form "n.n." For example, VSE/POWER Version 2, Release 3 is shown as VSE/POWER 2.3.
>
> If not stated otherwise, information given for the IBM 3800 Printing Subsystem applies also to the IBM 3200 Printing Subsystem.

CONTENTS

FIGURES

SUMMARY OF CHANGES

For a complete list of items that are new in this and the preceding release of version 2 of VSE/POWER, see VSE/POWER Installation and Operations Guide.

CHAPTER 1. JOB ACCOUNTING, SEGMENTATION, IBM 4248 SUPPORT

This chapter describes:

- The use of job accounting:

  Collecting accounting information for jobs running under the control of VSE/POWER. For details, see "Job Accounting."

- How to define output segmentation:

  Staging print or punched output of a job. This enables VSE/POWER to start processing a job's output before all processing for the job is finished. For details, see "Output Segmentation" on page 24.

The chapter also contains a short section on user-written channel programs for the IBM 4248 printer. For details, see "VSE/POWER Support of the IBM 4248" on page 27.

Throughout this chapter, the information and specifications given for the IBM 3800 apply also to the IBM 3200 if not specifically stated otherwise.

JOB ACCOUNTING

To have VSE/POWER job accounting support available, assemble and link the POWER generation macro with ACCOUNT=YES specified in the macro. VSE/POWER needs some additional processor and virtual storage, it needs disk space to accommodate the VSE/POWER account file. For information about these requirements, refer to Chapter 2: Planning and Installation of VSE/POWER Installation and Operations Guide.

VSE/POWER automatically collects job accounting information for each partition that runs under VSE/POWER control. VSE/POWER stores this information for each job step in chronological order in the account file. In this file, and also on tape or disk if the file was saved, the records are stored sequentially; they are in variable unblocked format on a CKD disk and on tape, they are in variable blocked format on an FBA disk.

Account-File-Full Condition: If the account file is full and a task of VSE/POWER must write another account record, this task waits until the operator issues a PACCOUNT command. Instruct your operator to do one of the following:

- Save the account records on tape (normally the preferred action).

- Save the account records on disk if a disk extent has been defined for this purpose.

- Have the contents of the account file spooled to the punch queue and punched out by starting a punch-writer task with class P.

If VSE/POWER is to spool the account file's contents, then each card-image (punch) record contains accounting information as follows:

Columns	Contents
1	Account-record identifier (field ACIDEN of the account record).
2-72	Data (bytes 0-70 of the account record) punched in the same positions as it appears in the account record, including the record identifier (not valid for continuation cards).
73-78	Record number of the account file.
79-80	Sequence number of continuation cards. One account record may require one or more punched cards.

Note: When shared spooling is used, each account record is preceded by a 16-byte account-record prefix as indicated by the SYSID operand of the PACCNT macro.

Processing Account Records: VSE/POWER produces various types of account records, and you may want to write a program of your own to process these records.

As an aid in writing such a program, you can use the PACCNT macro. The macro requests a DSECT to be assembled into your program for one, a selected number, or all types of account records.

The following types of account records are supported (the account-record identifier is in position 43 of each record):

- Execution account record (for its layout, see Figure 2 on page 7).

- List account record (for its layout, see Figure 3 on page 9).

- Network account record (for its layout, see Figure 4 on page 11).

- Punch account record (for its layout, see Figure 5 on page 12).

- Reader account record (for its layout, see Figure 6 on page 14).

- Transmitter/receiver account record (for its layout, see Figure 7 on page 15).

- RJE,BSC account record (for its layout, see Figure 8 on page 16).

- RJE,SNA account record (for its layout, see Figure 9 on page 17).
- System-up account record (for its layout, see Figure 10 on page 18).
- Spool-access connect account record (for its layout, see Figure 11 on page 19).
- Spool-access operation account record (for its layout, see Figure 12 on page 20).

You can cause additional information to be added to each execution account record by writing a routine that makes use of the PUTACCT macro. Link this routine as phase $JOBACCT in your system's sublibrary IJSYSRS.SYSLIB. This causes the IBM supplied dummy phase named $JOBACCT to be overwritten. For guidance how to write the routine, refer to the publication VSE/Advanced Functions, System Management Guide; an example for the use of the PUTACCT macro is given under "Example of the PUTACCT Macro" on page 23.

The remainder of this section describes the PACCNT macro, shows the layout of the VSE/POWER provided account records, and discusses the PUTACCT macro.

PACCNT Macro: Account-Record DSECT Generation

The macro causes a DSECT of one, a selected number, or all VSE/POWER-built account records to be assembled into your program. For most account-record types, the generated DSECT includes a fixed (header) part and a variable part. In your program, you would examine the record identifier and then work with the labels that apply to the specific record type. Give the generated DSECT a reading before you start coding a program using this DSECT.

For the format of account records when saved in punched cards, refer to "Account-File-Full Condition" on page 1.

Format of the Macro

The format of the macro as shown below does not include the continuation character, which you may have to code in column 72.

Name	Operation	Operands
[name]	PACCNT	[ALL={NO|YES}]
		[,BSC={NO|YES}]
		[,EXEC={NO|YES}]
		[,LIST={NO|YES}]
		[,PNET={NO|YES}]
		[,PUNCH={NO|YES}]
		[,READER={NO|YES}]
		[,RECV={NO|YES}]
		[,SNA={NO|YES}]
		[,SYS={NO|YES}]
		[,SYSID={NO|YES}]
		[,TRANS={NO|YES}]
		[,XCONN={NO|YES}]
		[,XSPOOL={NO|YES}]

For name (in the name field), specify the label you want to use for referring to the DSECT for the account record(s).

Description of Operands

ALL={NO|YES}
Specify ALL=YES to have DSECTs generated for all account records. If the account prefix is required, then specify also SYSID=YES. Any other specification that you may supply is ignored.

BSC={NO|YES}
Specify BSC=YES to have a DSECT generated for an RJE,BSC account record (for its layout, see Figure 8 on page 16).

EXEC={NO|YES}
Specify EXEC=YES to have a DSECT generated for an execution account record (for its layout, see Figure 2 on page 7).

LIST={NO|YES}
Specify LIST=YES to have a DSECT generated for a list account record (for its layout, see Figure 3 on page 9).

PNET={NO|YES}
Specify PNET=YES to have a DSECT generated for a PNET network account record (for its layout, see Figure 4 on page 11).

PUNCH={NO|YES}
Specify PUNCH=YES to have a DSECT generated for a punch account record (for its layout, see Figure 5 on page 12).

READER={NO|YES}
Specify READER=YES to have a DSECT generated for a reader account record (for its layout, see Figure 6 on page 14).

RECV={NO|YES}
Specify RECV=YES to have a DSECT generated for a transmitter/receiver account record (for its layout, see Figure 7 on page 15).

SNA={NO|YES}
Specify SNA=YES to have a DSECT generated for an RJE,SNA account record (for its layout, see Figure 9 on page 17).

SYS={NO|YES}
Specify SYS=YES to have a DSECT generated for a system-up account record (for its layout, see Figure 10 on page 18).

SYSID={NO|YES}
Specify SYSID=YES to have the 16-byte account-record prefix generated into and at the beginning of the account-record DSECT (for its layout, see Figure 1 on page 6).

TRANS={NO|YES}
Specify TRANS=YES to have a DSECT generated for a transmitter/receiver account record (for its layout, see Figure 7 on page 15).

XCONN={NO|YES}
Specify XCONN=YES to have a DSECT generated for a spool-access connect account record (for the record's layout, see Figure 11 on page 19).

XSPOOL={NO|YES}
Specify XSPOOL=YES to have DSECTs generated for a spool-access operation account record (for the layout of this record, see Figure 12 on page 20).

Legend: C = character (alphameric); B = binary;

Field Name	Description	Field Type & Length
ACSYSID	System identifier.	CL1
ACTYPE	Record type (character) and version (binary).	BL2
ACCOMP	Component identifier: 5666273b, where b=blank.	CL8
	Reserved.	CL5

Figure 1. Account-Record Prefix

Layout of Account Records

VSE/POWER builds one execution account record for each VSE job step (each time a // EXEC or /& statement occurs). If a job or job step is canceled, VSE/POWER's statistics reflect the processing up to the point of this cancellation. The record can be up to 2008 bytes long.

The same execution account record is used by VSE/POWER itself when the operator has issued a PEND command. In this VSE/POWER execution account record, certain fields are set to zero (see below).

Legend: C = character (alphameric); B = binary; P = packed decimal;
E.A.R. = execution account record

Field Name	Description	Field Type & Length
ACDATE	Date in the format as defined for the system.	CL8
ACSTRT	Start time of job step (0hhmmssf, where f = sign).	PL4
ACSTOP	Stop time of job step (0hhmmssf, where f = sign). This time may be higher than the time logged on the console; it accounts for VSE/POWER job termination.	PL4
ACUSER	16 bytes of user information from * $$ JOB card.	CL16
ACNAME	Current VSE/POWER job name or AUTONAME.	CL8
ACNUMB	Job number assigned by VSE/POWER.	BL2
ACIDEN	Record identifier (E).	CL1
ACCANC	VSE/POWER cancel code: X'10' = Normal end of VSE/POWER job or task. The associated VSE job(s) may have been canceled by the system nevertheless. X'30' = PSTOP command was issued. The code is not stored in the account record if the EOJ option was specified in the PSTOP command. X'40' = PFLUSH command was issued. X'70' = The job was canceled due to an I/O error.	BL1
	Reserved.	CL4
EXFRM	FROM remote ID.	BL1
	Reserved	BL1
EXICL	Class.	CL1
EXIPR	Priority.	CL1
EXNLN	Number of lines spooled (zero for VSE/POWER-E.A.R.).	BL4
EXNCD	Number of cards spooled (zero for VSE/POWER-E.A.R.).	BL4
EXNPG	Number of pages spooled (zero for VSE/POWER-E.A.R.).	BL2
EXSIO	Length of SIO table.	BL2
EXTAC	Length of total execution account record.	BL2
	Reserved.	CL4
EXOJ#	Original job number, if one exists.	BL2
EXXNODE	Name of execution node.	CL8
EXFRNO	Name of FROM (originating) node.	CL8
EXFRUS	Identifier of originating user.	CL8

Figure 2 (Part 1 of 2). Execution Account Record

Legend: C = character (alphameric); B = binary; P = packed decimal

Field Name	Description	Field Type & Length
EXDJOB	VSE job name from the // JOB card.	CL8
EXDUSER	16 bytes of user information from the // JOB card. On shutdown, the field contains: VSE/POWER-E.A.R., where E.A.R. = execution account record.	CL16
EXPID	Partition ID in EBCDIC format.	CL2
EXDCANC	VSE cancel code (see VSE/System Package, Messages and Codes).	BL1
EXTYPE	Type of record; S = job step, L = last step.	CL1
EXJDUR	Duration of job step (in 300ths of a second)	BL4
EXPHASE	Phase name, taken from the // EXEC card.	CL8
EXPASZ	Number of pages multiplied by 2K bytes	BL4
EXCPUTM	Processor time in 300ths of a second. This is the actual time used by a job or job step in the system.	BL4
EXOVHTM	Overhead time in 300ths of a second. This is the time needed for activities that cannot be charged to a specific program or partition. For example, the time for calling a routine, for error recovery, or from the start of the $JOBACCT routine to the processing of the EXEC statement. All SVC processing is counted as active processor time for the job or job step. Overhead time is distributed over the number of active partitions.	BL4
EXALLTM	Total system wait time in 300ths of a second. This time is divided by the number of active partitions based on the percentage of used processor time.	BL4
EXSIOTB	SIO tables. Six bytes per device defined to the system during system start-up: bytes 0 and 1 = 0cuu; bytes 2 through 5 = count of SIOs in current job step. VSE/POWER updates the SIO tables in the execution account record with the number of I/Os it has intercepted for spooling purposes.	BL6 for each SIO entry
EXSIOTB+n	Set by the system to X'20'. n = total length of the SIO tables (EXSIO)	
EXSIOTB+m	User account information (provided via a PUTACCT macro). m = EXSIOTB+n+1.	

Figure 2 (Part 2 of 2). Execution Account Record

VSE/POWER builds a list account record for each queue entry that is processed by a list task.

Legend: C = character (alphameric); B = binary; P = packed decimal

Field Name	Description	Field Type & Length
ACDATE	Date in the format as defined to the system.	CL8
ACSTRT	Start time of list output (0hhmmssf, where f = sign).	PL4
ACSTOP	Stop time of list output (0hhmmssf, where f = sign).	PL4
ACUSER	16 bytes of user information from the * $$ JOB or the * $$ LST statement.	CL16
ACNAME	VSE/POWER job name from the * $$ JOB or the // JOB statement.	CL8
ACNUMB	Job number assigned by VSE/POWER (same as that of the associated reader queue entry).	BL2
ACIDEN	Record identifier (L).	CL1
ACCANC	VSE/POWER cancel code: X'10' = Normal end of VSE/POWER job or task - The associated VSE job(s) may have been canceled by the system nevertheless. X'30' = PSTOP command was issued. The code is not stored in the account record if the EOJ option was specified in the PSTOP command. X'40' = PFLUSH command was issued. X'70' = The job was canceled due to an I/O error.	BL1
	Reserved.	CL1
LSTADR	Printer or RJE-line address (cuu), SNA, or GSP.	CL3
LSTFRM	FROM remote ID.	BL1
LSTTO	TO remote ID.	BL1
LSTOCL	Printed output class.	CL1
LSTOPR	Printed output priority number.	CL1
LSTNUM	Number of lines printed (see also LSTEXR).	BL4
LSTTRK	Number of DBLK groups for output storage. The field is set to zero if the list output was spooled to tape.	BL2

Figure 3 (Part 1 of 2). List Account Record

Legend: C = character (alphameric); B = binary; P = packed decimal

Field Name	Description	Field Type & Length
LSTSUF	Job suffix (segment) number assigned by VSE/POWER. If: X'00' - The only segment for a job. X'82' or higher - The last segment for a job; the seven low-order bits give the number of segments.	BL1
LSTCOP	Number of printed copies. (If more than one, the statistics are totals for all copies).	BL1
LSTFOR	Print-forms identification.	CL4
LSTEXR	Number of extra records printed due to a restart, a PSETUP request, separator pages, or extra copies.	BL4
LSTPAG	Number of pages printed (skips to channel 1 or page filled; see also LSTEXP).	BL2
LSTEXP	Number of extra pages printed due to PRESTART, PSETUP, separator cards, or extra copies.	BL2
LSTFLSH	Flash identifier (applies only to 3800 printer).	CL4
LSTCPYG	Copy groupings (applies only to 3800 printer).	CL8
LSTNODE	Name of own node (system) in the network.	CL8
LSTTOUS	Destination-user (TO) identification.	CL8
LSTFRNO	Name of originating (FROM) node.	CL8
LSTFRUS	Originating-user (FROM) identification.	CL8
LSTOJ#	Original job number, if one exists.	BL2

Figure 3 (Part 2 of 2). List Account Record

VSE/POWER builds a network account record for an existing communication path when a session via this path is terminated. The record contains information about all activities during the session.

Legend: C = character (alphameric); B = binary; P = packed decimal

Field Name	Description	Field Type & Length
NETDTE	Date in the format as defined for the system.	CL8
NETSGN	Sign-on time (0hhmmssf, where f = sign).	PL4
NETSGF	Sign-off time (0hhmmssf, where f = sign).	PL4
NETNODE	Identifier of the connected node.	CL8
NETNPAS	Node password.	CL8
NETPSW	Line password.	CL8
NETICNT	Invalid responses per session.	BL2
NETIDEN	Account-record identifier (N)	CL1
NETTERM	ACF/VTAM cancel code: X'02' = Normal ACF/VTAM shutdown. X'04' = Abnormal end of ACF/VTAM. X'08' = An internal error occurred X'10' = A line error occurred or a session was terminated. X'20' = A time-out occurred. X'40' = A remote SIGNOFF occurred. X'80' = Cancel on operator request (one of the commands PSTOP and PEND.	BL1
	Reserved.	BL1
NETLAD	Line address or SNA.	CL3
NETTRAN	Transmission count (of buffers) per session.	BL4
	For PNET support using BSC	
NETTCNT	Timeout count per session.	BL2
NETERR	Error count per session.	BL2
	For PNET support using SNA	
NETRCVE	Buffers received during session	BL4
NETSOD	Sign-off date.	CL8

Figure 4. Network Account Record

Account Record - Punch

VSE/POWER builds a punch account record for every punch-queue entry that is processed by a punch task.

Legend: C = character (alphameric); B = binary; P = packed decimal

Field Name	Description	Field Type & Length
ACDATE	Date in the format defined to the system.	CL8
ACSTRT	Start time of punch output (0hhmmssf, where f = sign).	PL4
ACSTOP	Stop time of punch output (see ACSTRT, above).	PL4
ACUSER	16 bytes of user information from the * $$ JOB card.	CL16
ACNAME	VSE/POWER job name from the * $$ JOB card.	CL8
ACJOBN	Job number assigned by VSE/POWER (same as that of associated reader queue entry).	BL2
ACIDEN	Account-record identifier (P).	CL1
ACCANC	VSE/POWER cancel code: X'10' = Normal end of VSE/POWER job or task - The associated VSE job(s) may have been canceled by the system nevertheless. X'30' = PSTOP command was issued. The code is not stored in the account record if the EOJ option was specified in the PSTOP command. X'40' = PFLUSH command was issued. X'70' = The job was canceled due to an I/O error.	BL1
	Reserved.	CL1
PUNADR	Punch device or RJE-line address (cuu), SNA, or GSP.	CL3
PUNFRM	FROM remote ID.	BL1
PUNTO	TO remote ID.	BL1
PUNOCL	Punched output class.	CL1
PUNOPR	Punched output priority number.	CL1
PUNNUM	Number of records punched (see also PUNEXR).	BL4
PUNTRK	Number of DBLK groups for output storage. The field is set to zero if the output was spooled to tape.	BL2

Figure 5 (Part 1 of 2). Punch Account Record

Legend: C = character (alphameric); B = binary; P = packed decimal

Field Name	Description	Field Type & Length
PUNSUF	Job suffix (segment) number assigned by VSE/POWER.	BL1
PUNCOP	Number of punched copies (if more than one, the statistics are the totals for all copies).	BL1
PUNFOR	Punch-forms identification.	CL4
PUNEXR	Number of additional cards punched due to restart, separator cards, or extra copies.	BL4
PUNNODE	Name of own node (system) in the network.	CL8
PUNTOUS	Destination-user (TO) identification.	CL8
PUNFRNO	Name of originating (FROM) node.	CL8
PUNFRUS	Originating-user (FROM) identification.	CL8
PUNOJ#	Original job number, if one exists.	BL2

Figure 5 (Part 2 of 2). Punch Account Record

VSE/POWER builds a reader account record for every VSE/POWER job submitted for spooling. Whether the queue entry has actually been queued is indicated by the VSE/POWER cancel code. VSE/POWER does not build reader account records for a writer-only partition.

Legend: C = character (alphameric); B = binary; P = packed decimal

Field Name	Description	Field Type & Length
ACDATE	Date in the format as defined to the system.	CL8
ACSTRT	Start time of read (0hhmmssf, where f = sign).	PL4
ACSTOP	Stop time of read (see ACSTRT, above).	PL4
ACUSER	16 bytes of user information from the * $$ JOB statement.	CL16
ACNAME	VSE/POWER job name from the * $$ JOB or the // JOB statement.	CL8
ACNUMB	Job number assigned by VSE/POWER.	BL2
ACIDEN	Record identifier (R).	CL1
ACCANC	VSE/POWER cancel code: X'10' = Normal end of VSE/POWER job or task - The associated VSE job(s) may have been canceled by the system nevertheless. X'30' = PSTOP command was issued. The code is not stored in the account record if the EOJ option was specified in the PSTOP command. X'40' = PFLUSH command was issued. X'60' = The job was canceled via RDREXIT. X'70' = The job was canceled due to an I/O error.	BL1
	Reserved.	CL1
RDRADD	Reader device or line address (cuu), SNA, or PSP for submission from a partition.	CL3
RDRFRM	FROM remote ID.	BL1
	Reserved.	BL1
RDRICL	Input class.	CL1
RDRIPR	Input priority number.	CL1
RDRNUM	Number of records read (including record added or deleted by a reader exit routine).	BL4
RDRTRK	Number of DBLK groups for input storage.	BL2
RDRNODE	Name of own node (system) in the network.	CL8
RDRFRUS	Originating-user (FROM) identification.	CL8

Figure 6. Reader Account Record

VSE/POWER builds a transmitter/receiver-account record for every job or output transmission via a connection or during a session.

Legend: C = character (alphameric); B = binary; P = packed decimal

Field Name	Description	Field Type & Length
ACDTE	Date in the format as defined to the system.	CL8
ACSTRT	Start time (0hhmmssf, where f = sign).	PL4
ACSTOP	Stop time (0hhmmssf, where f = sign).	PL4
ACUSER	User information.	CL16
ACNAME	Job name.	CL8
ACNUMB	Job number.	BL2
ACIDEN	Record identifier (V = receiver; M = transmitter).	CL1
ACCANC	VSE/POWER cancel code: X'10' = Normal end of VSE/POWER transmitter or receiver task. X'30' = PSTOP command was issued. The code is not stored in the account record if the EOJ option was specified in the PSTOP command. X'40' = PFLUSH command was issued. X'60' = The job was canceled via RDREXIT. X'70' = The job was canceled due to an I/O error. X'80' = The job or output transmission was canceled due to a receiver-task stop at the other end.	BL1
	Reserved.	CL1
NACLAD	Line address (cuu) or SNA.	CL3
NACQTYP	Queue type (R = reader; L = list; P = punch).	CL1
	Reserved.	BL1
NACCLAS	Class of job output.	CL1
NACPR	Priority.	CL1
NACCNTD	Data record count.	BL4
NACORGJ#	Original job number from job reader.	BL2
NACSUF	Job suffix (segment) number.	CL1
NACCOP	Number of copies.	BL1
NACCNTC	Control record count.	BL2
	Reserved.	BL2
NACON	Name of originating node.	CL8
NACOUS	Name (user identifier) of remote originator.	CL8
NACTN	Name of destination node.	CL8
NACTUS	Destination-user identifier.	CL8
NACCURR	Current (own VSE) node name.	CL8
NACADJ	Adjacent node name.	CL8

Figure 7. Transmitter- or Receiver-Account Record

Account Record - RJE,BSC

VSE/POWER builds an RJE,BSC account record for an RJE,BSC user session when it processes a sign-off or when a line stop occurs.

Legend: C = character (alphameric); B = binary; P = packed decimal

Field Name	Description	Field Type & Length
BSCDTE	Date in the format as defined to the system.	CL8
BSCSGN	SIGNON time (0hhmmssf, where f = sign).	PL4
BSCSGF	SIGNOFF time (0hhmmssf, where f = sign).	PL4
BSCUSE	16 bytes of user information from the SIGNON command.	CL16
BSCPAS	Line password.	CL8
BSCIRS	Number of invalid responses during transmission (see the note below).	BL2
BSCIDN	Record identifier (T).	CL1
BSCSFC	SIGNOFF code (any combination of the codes may occur): X'01' = Normal SIGNOFF. X'02' = SIGNOFF forced due to PSTOP cuu. X'04' = SIGNOFF forced due to excessive idle time. X'08' = SIGNOFF forced due to unrecoverable I/O error. X'10' = SIGNOFF forced due to PEND or PSTOP cuu,EOJ. X'20' = SIGNOFF forced by lack of processor storage. X'40' = SIGNOFF forced due to PSTOP cuu,FORCE. X'80' = SIGNOFF forced due to line stop at last I/O.	BL1
BSCTEC	Terminal (work-station) error count.	BL1
BSCLAD	Line address.	CL3
BSCRID	Remote identifier.	BL1
	Reserved.	CL1
BSCTRAN	Transmission count per session (see the note below).	BL2
BSCTCNT	Timeout count per session (see the note below).	BL2
BSCERR	Error count per session (see the note below).	BL2
BSCSOD	SIGNOFF date (mmddyy).	CL6

Comparing fields BSCTRAN and BSCTCNT gives an indication of idle time per session. Comparing fields BSCTRAN, BSCTCNT, and BSCERR gives an indication of line quality.

Figure 8. RJE,BSC Account Record

VSE/POWER builds an RJE,SNA account record when an RJE,SNA user session ends.

Legend: C = character (alphameric); B = binary; P = packed decimal

Field Name	Description	Field Type & Length
SNADTE	Date in the format as defined to the system.	CL8
SNASGN	SIGNON time (0hhmmssf, where f = sign).	PL4
SNASGF	SIGNOFF time (0hhmmssf, were f = sign).	PL4
SNAUSE	16 bytes of user information from the SIGNON command.	CL16
SNALUN	Logical unit name.	CL8
	Number of invalid responses during transmission.	CL2
SNAIDEN	SNA record identifier (S).	CL1
SNATERM	Session termination code: X'01' = normal termination (LOGOFF or SIGNOFF) X'02' = abnormal termination	BL1
SNARID	Remote identifier.	BL4

Figure 9. RJE,SNA Account Record

VSE/POWER builds a system-up account record on completion of VSE/POWER start-up.

Legend: C = character (alphameric); B = binary; P = packed decimal

Field Name	Description	Field Type & Length
PWRDTE	Date in the format as defined to the system.	CL8
PWRSGN	Start-up time.	PL4
	Reserved.	BL4
PWRVER	Version/Modification level	CL4
PWRLEV	Level identifier.	CL4
PWRPARSZ	Partition size.	BL4
PWRGETSZ	GETVIS size.	BL4
PWRRELSZ	Reserved processor (real) storage size.	BL4
PWRPART	Partition identifier (BG or Fn).	CL2
PWRFLAG	Feature flags.	CL4
PWRIDEN	Record identifier (U).	CL1
PWRDXTN	Number of data file extents.	BL1
PWRDTRK	Number of tracks/blocks in the data file.	BL4
PWRQTRK	Number of tracks/blocks in the queue file.	BL4
PWRATRK	Number of tracks/blocks in the account file.	BL4

Figure 10. VSE/POWER System-Up Account Record

VSE/POWER builds a spool-access connect account record whenever an established communication path is terminated, normally or abnormally.

Legend: C = character (alphameric); B = binary; P = packed decimal

Field Name	Description	Field Type & Length
XCODATE	Date in the format defined to the system.	CL8
XCOSTRT	Connection start time (0hhmmssf, where f = sign).	PL4
XCOSTOP	Connection stop time (0hhmmssf, where f = sign).	PL4
XCOAPPL	XPCC application identifier.	CL8
XCOMSG#	Number of messages returned in response to a CTL or PUT request.	BL4
XCOCTL#	Number of CTL requests.	BL4
XCOTERM	Connection-termination code:	BL1
XCOTCOK	X'01' = Normal end of communication.	
XCOTCPD	X'02' = Termination because of a PEND command.	
XCOTCPP	X'04' = Termination because of a PSTOP command (but (not if FORCE is specified).	
XCOTCAT	X'08' = Abnormal end by user application	
XCOTCUE	X'10' = Severe error in the application program	
XCOTCKL	X'20' = Termination because of a PSTOP command (if FORCE is specified).	
XCOTCSE	X'40' = System or VSE/POWER failure.	
	Reserved.	BL1
XCODEVN	Device name (as defined to the device-owning subsystem.	CL8
XCOIDEN	Account-record identifier (C)	CL1

Figure 11. VSE/POWER Spool-Access Connect Account Record

VSE/POWER builds a spool-access operation account record for a PUT or a GET service when the processing for a queue entry is finished. If data is added to an appendable output, VSE/POWER builds an account record of this type each time a program finishes appending data to this output.

No accounting is performed for CTL requests or for output queue entries that are held in the queue with a disposition of X (because of an abnormal termination of VSE/POWER).

Legend: C = character (alphameric); B = binary; P = packed decimal

Field Name	Description	Field Type & Length
XSPDATE	Date in the format as defined to the system.	CL8
XSPSTRT	Start time of processing (0hhmmssf, where f = sign).	PL4
XSPSTOP	Stop time of processing (0hhmmssf, where f = sign).	PL4
XSPUSER	16 bytes of user information (field SPLDUI of the PWRSPL DSECT).	CL16
XSPNAME	Name of job (or report).	CL8
XSPNUMB	Job number as assigned by VSE/POWER.	BL2
XSPIDEN	Account-record identifier (X)	CL1
XSPCANC	VSE/POWER cancel code: X'10' = Normal end of VSE/POWER job or task - The associated VSE job(s) may have been canceled by the system nevertheless. X'40' = A PFLUSH command was issued. X'50' = A purge request (during a queue entry retrieval) or a PDELETE command was issued. X'90' = A quit request was issued. X'A0' = The operation was terminated because a severe error occurred or the system failed to maintain the communication path. X'B0' = A Close request was issued. X'C0' = Canceled due to lack of disk space. X'D0' = A 'quit-and-lock' request was issued.	BL1
	Reserved.	CL1
XSPREQT	Request type (G = GET request; P = PUT request).	CL1
XSPQUID	Queue type (R = reader; L = list; P = punch).	CL1
XSPJSUF	Job-suffix (output segment) number.	BL1
XSPCLSS	Class.	CL1
XSPPRIO	Priority.	CL1
XSPDISP	Disposition.	CL1
XSPCOPY	Number of copies (output only).	BL1
XSPCPYG	Copy groupings (3800 output only).	BL8

Figure 12 (Part 1 of 2). VSE/POWER Spool-Access Operation Account Record

Legend: C = character (alphameric); B = binary; P = packed decimal

Field Name	Description	Field Type & Length
XSPTRK#	Number of DBLK groups occupied on disk (see Note 1, below).	BL2
XSPOJ#	Original job number.	BL2
XSPREC#	Number of records. The value includes the control record, even if a spool-access user has not specified CTLREC=YES in the applicable PWRSPL macro. SPL records returned by VSE/POWER are not included in this record count. See also Note 1, below.	BL4
XSPEXR#	Number of extra records.	BL4
XSPLNE#	Total number of lines or cards (output only); see also Notes 1 and 2, below.	BL4
XSPEXL#	Number of extra lines or cards because of separator pages or cards, or because of records repeatedly as a result of a restart (applies only to GET requests).	BL4
XSPPGE#	Total number of pages (output only); see also Notes 1 and 3, below.	BL4
XSPEXP#	Number of extra pages such as separator pages or pages passed repeatedly as a result of a restart (applies only to GET for output).	BL4
XSPFORM	Forms identification (applies to output only).	CL8
XSPFLSH	Flash identification (applies to 3800 output only).	CL4
XSPTONM	Name of destination node.	CL8
XSPTOUS	Destination-user identifier.	CL8
XSPRQUS	Requesting-user identifier.	CL8
XSPRQAP	Requesting XPCC application identifier.	CL8
XSPNODE	Name of your own node.	CL8

1. The count applies to and is shown for appendable output only.

2. The line-number count is set to the record count if the queue entry's record format is SCS, 3270 data stream, BMS, or escape mapping.

3. The total page count is not meaningful for a list queue entry containing data in the BMS mapping or the 3270 data stream format. For output of this type, each record is considered to be a page. If the queue entry contains data in the SCS or the escape mapping format, the total page count is not meaningful either and, therefore, set to zero.

Figure 12 (Part 2 of 2). VSE/POWER Spool-Access Operation Account Record

PUTACCT Macro: Adding User Information to Account Records

The macro, which you can use in your $JOBACCT routine, adds additional information to the end of the execution account record. VSE/POWER calls your $JOBACCT routine at the end of each job or job step.

Before you issue the PUTACCT macro in your $JOBACCT routine, save registers 0 and 1. These registers are used and overwritten by VSE/POWER.

VSE/POWER ignores the macro if job accounting has not been defined for VSE/POWER control-table generation.

Format of the Macro

Name	Operation	Operands
[name]	PUTACCT	(regl),(reg2)

Description of Operands

(regl),(reg2)
For the operands regl and reg2, specify two different general registers, but not registers 0 and 1. When you issue the macro, the registers must contain the following:

regl The address of the area that contains the additional information.

reg2 The length of the above mentioned ar3a.

The maximum length of the area may not exceed 2,008 bytes minus:

1. Eight bytes for the control field.
2. The length of the execution account record as set up by VSE/POWER.

An example for using the macro is given on the next page.

Example of the PUTACCT Macro

The coding example below inserts additional information behind the VSE/POWER execution account records:

```
           ... ... ...
           COMRG REG=R4          GET PARTITION COMMUNICATION REGION
           USING CMRG,R4         DECLARE ADDRESSABILITY
           TM    POWFLG1,X'80'   ACCOUNT SUPPORT FOR THIS PARTITION
           DROP R4
           BNO   EXIT            BRANCH IF NOT
           LA    R2,ADAC         ADDRESS ADDITIONAL INFORMATION
           LA    R3,L'ADAC       LENGTH ADDITIONAL INFORMATION
           PUTACCT (R2),(R3)     PASS INFORMATION TO VSE/POWER
  EXIT     DS    0H
           BR    RE              RETURN TO $JOBCTLN
           ... ... ...
  ADAC     DC    C'ADDITIONAL ACCOUNT INFORMATION'
  R2       EQU   2               REGISTER 2
  R3       EQU   3               REGISTER 3
  R4       EQU   4               REGISTER 4
  RE       EQU   14              REGISTER 14
           ... ... ...
  CMRG     MAPCOMR
           ... ... ...
| +POWFLG1 DS    X
           ... ... ...
           END
```

OUTPUT SEGMENTATION

VSE/POWER job output can be segmented, that is, part of the output from a job can be printed or punched before the entire job is finished.

Four types of segmentation are possible based on the event that initiates the segmentation:

- Program-driven output segmentation:

  In your application program you may use the VSE/POWER SEGMENT macro or the VSE LFCB macro to separate the output. The LFCB macro causes segmentation before loading the new FCB.

  If your output is directed to an IBM 3800, you may use a VSE SETPRT job control statement requesting a printer setup to cause segmentation. These are setup requests that require operator intervention, change copy grouping or the FCB image, or specify a copy number greater than one.

The following types of output segmentation are described in VSE/POWER Installation and Operations Guide:

- Count-driven output segmentation
- Data-driven output segmentation
- Multivolume tape segmentation

SEGMENT Macro: Control Output Segmentation

The macro can be used for controlling output segmentation for a job running in a VSE/POWER-controlled partition. You can use it for the specification of new output controls that are to apply to the next segment.

VSE/POWER assigns a new job number for the second and each subsequent SEGMENT request of your program.

Before using the macro, save your registers 0 and 1 because they are used and overwritten by VSE/POWER. Register 15 contains the return code passed by VSE/POWER on completion of the segment request.

Format of the Macro

Name	Operation	Operands
[name]	SEGMENT	DEVADDR=SYSxxx [,FORMS=formnumber] [,JECL={addr|(reg)}] [,NAME={name|(reg)}]

Description of Operands

DEVADDR=SYSxxx

For SYSxxx, specify the system or programmer logical unit assigned to the device on which the segmentation is to occur.

Your device specification in this operand must match your specification in the

LST operand of * $$ LST for list output, or
PUN operand of * $$ PUN for punch output

if you supplied a device specification in that statement.

FORMS=formnumber

For formnumber, specify the new one- to four-character form number which VSE/POWER is to use in its forms-mount messages.

VSE/POWER sets the form number to blanks if you:

1. Omit the operand, and
2. Do not supply a form number in a new * $$ LST or * $$ PUN statement pointed to by your specification in the JECL operand.

JECL={address|(reg)}

The operand points to a 71-byte area that contains one of the following JECL statements: * $$ LST, * $$ PUN, or * $$ JOB. These statements are described in the VSE/POWER Installation and Operations Guide.

For address, specify the area's label in your program.

For reg, if you choose register notation, specify the register that contains the address of the area.

If you omit the specification of an * $$ LST or * $$ PUN statement, then default spooling values will be set for the new segment (regardless of any previously established values); therefore pass only the operands needed to change default values which do not meet your requirements.

Notes:

1. The macro causes new values to be set for the new segment. However, passing an * $$ JOB statement with the NAME operand causes the currently processed segment to be renamed. The statement should therefore be passed by a separate SEGMENT macro after an * $$ LST or * $$ PUN statement was passed. You should pass an * $$ JOB statement only to provide new user information.

2. If you include an LST or a PUN operand in the passed * $$ LST or * $$ PUN statement, the specification is ignored.

3. If output segmentation is requested for output on an IBM 3800 printer, VSE/POWER uses the default printer setup for the new segment. If this is not desirable, supply an * $$ LST statement defining the desired printer setup. After having issued the SEGMENT macro, you may, in your program, issue a SETPRT macro requesting the proper printer setup.

NAME={name|(reg)}
For name, specify a one- to eight-character name that you want VSE/POWER to assign to the new segment. If you omit this operand, VSE/POWER uses the name by which the job was placed into the input queue.

If you use register notation, the specified register must point to an eight-byte field containing the name of the segment.

Example of the SEGMENT Macro

```
                                              Column 72 ─────┐
          ... ... ...                                        |
         LA      R2,LSTCARD                                  V
         SEGMENT DEVADDR=SYSLST,JECL=(R2),                   C
                 NAME=TESTOUT
          ... ... ...
LSTCARD  DC    CL71'* $$  LST FNO=ACB1,DISP=H,PRI=1'
          ... ... ...
```

Return Codes from the SEGMENT Macro

Successful completion of the SEGMENT macro is indicated to the problem program by a return code of 0 in register 15. If the operation fails, register 15 contains one of the return codes listed below.

Code **Meaning**

X'04' One of the following:

- The device specified in the DEVADDR operand is not being spooled by VSE/POWER.
- VSE/POWER is not active.
- The partition in which your program is running is not under control of VSE/POWER.
- The spooled device is used for output with a disposition of N.
- The passed JECL statement was not one of:

```
* $$ JOB
* $$ LST
* $$ PUN
```

X'08' VSE/POWER cannot accept the JECL statement because either:

- The partition was not started as a multitasking partition and the partition is waiting for work, or
- The partition was started as a multitasking partition and is waiting for work, but no JECL statement was submitted for the specified device.

VSE/POWER SUPPORT OF THE IBM 4248

A program that writes to an IBM 4248 printer operating in native mode can run under control of VSE/POWER. In general, there is no need for you to change your programs. VSE/POWER handles IBM 4248-specific I/O requests as described in VSE/POWER Installation and Operations Guide.

As far as user-written channel programs are concerned, some of the IBM 4248-specific I/O commands cannot be processed such that they achieve the expected result. These commands are listed in Figure 13 on page 28.

Command		Action by VSE/POWER During	
Name	Code	Job Execution	Printing Spooled Output
Read Band ID	X'0A'	Returns the requested bytes with all bits set to zero.	Ignores the command.
Execute Order	X'33'	Spools the command, except a Purge Buffered Data order.	Ignores the command if: - Horizontal-copy printing is not set in the FCB. - The command is a Purge Buffered Data order.
Load FCB	X'63'	Spools the command, including the FCB image.	Passes the command and the image to the printer. However, VSE/POWER loses control over the printer's FCB.
Sense ID	X'E4'	Returns the requested 7-byte device identification.	Ignores the command.
Sense Intermediate Buffer	X'14'	Returns the requested bytes with all bits set to zero.	Ignores the command.
Verify Band ID	X'F3' or X'FB'	Returns the requested bytes with all bits set to zero.	Ignores the command.
Printer control commands not listed here are handled by VSE/POWER in the same way as in the past.			

Figure 13. VSE/POWER Action for IBM 4248-Specific I/O Commands

CHAPTER 2. SPOOL-ACCESS SUPPORT

The spool-access support allows a program running under or outside the control of VSE/POWER to access VSE/POWER services. A program using the support can, for example:

- Retrieve queue entries from the local VSE/POWER queues.
- Submit jobs or output data for spooling to the VSE/POWER queues.
- Submit control requests or pass VSE/POWER commands (such as PALTER, PDISPLAY, PHOLD, PXMIT) to control the handling of specific queue entries.

Normally, IBM supplied components make use of this kind of support, transparent to you. If you do not plan to implement applications that require this support, then there is no need for you to read this chapter.

Prerequisites: To understand this chapter, you should be familiar with the conventions and use of assembler language and the coding of macros; you should also have a good knowledge of the operational concepts of VSE/POWER and its use.

Also, it might be helpful to have a copy of the VSE/POWER Installation and Operations Guide readily available for reference.

Program Compatibility: VSE/POWER-access support has been available in the past with the XECBTAB-based macros CTLSPOOL, GETSPOOL, and PUTSPOOL. The support provided by these macros (referred to as SPOOL-macro support) continues to be available to ensure program compatibility. A description of these macros is given in Appendix B. VSE/POWER's spool-access support and the previously available SPOOL-macro support can be used side by side in one program.

Advantages: Using the new spool-access support is of advantage because, for example:

- It allows more than one program to access VSE/POWER at the same time.
- It enables VSE/POWER to return a variety of control data if a display of status information is requested.
- It allows a program to submit data, such as a report or a document, for spooling into an output queue.
- It supports checkpointing and restarting during both retrieval of spool records and submission of records for spooling.

- It allows communication between VSE/POWER and a partition in any address space.

IBM recommends that you use this new spool-access support in any application program that is to be changed or redesigned to include VSE/POWER spool-access services.

This chapter briefly discusses the operational concept of the spool-access support; it describes how to use the support; it includes a description of the applicable macros and their operands. The chapter includes a sample program at the end.

CONCEPT

To use the available VSE/POWER-access services, your program must:

1. Set up a communication path to VSE/POWER
2. Issue one or more requests to obtain the desired spool-access service.
3. Remove the existing communication path when there is no further need for access services.

You accomplish this by using the macros whose purposes are summarized below. For a description of these macros, refer to "Description of the Spool-Access Support Macros" on page 97.

XPCCB Control-block generation macro

The macro builds the control block (called XPCCB) needed to service the request initiated by an XPCC macro.

MAPXPCCB DSECT generation macro

The macro builds a DSECT for access to an XPCCB. In this chapter, references to XPCCB related fields or codes use the mnemonics that you find also in the generated DSECT.

XPCC The actual access-request macro

You use this macro in your program wherever there is a need for a spool-access service by VSE/POWER. Normally, you issue a number of such requests for a queue entry retrieval or a job or output submission; it may be just one request for a control-type service.

PWRSPL Spool parameter list (SPL) macro

The macro builds a parameter list. The list is used to pass, to VSE/POWER, the control information needed to render the desired access service. VSE/POWER requires this list when your program issues the first (or only) request for the access service.

On request, the macro generates a DSECT of the SPL. In this chapter, references to SPL-related fields or codes use the mnemonics that you find also in the generated DSECT.

Figure 14 on page 32 shows how the macros XPCC, XPCCB, and PWRSPL, relate to each other; it shows how the associated control blocks and areas are used for setting up an access to VSE/POWER services.

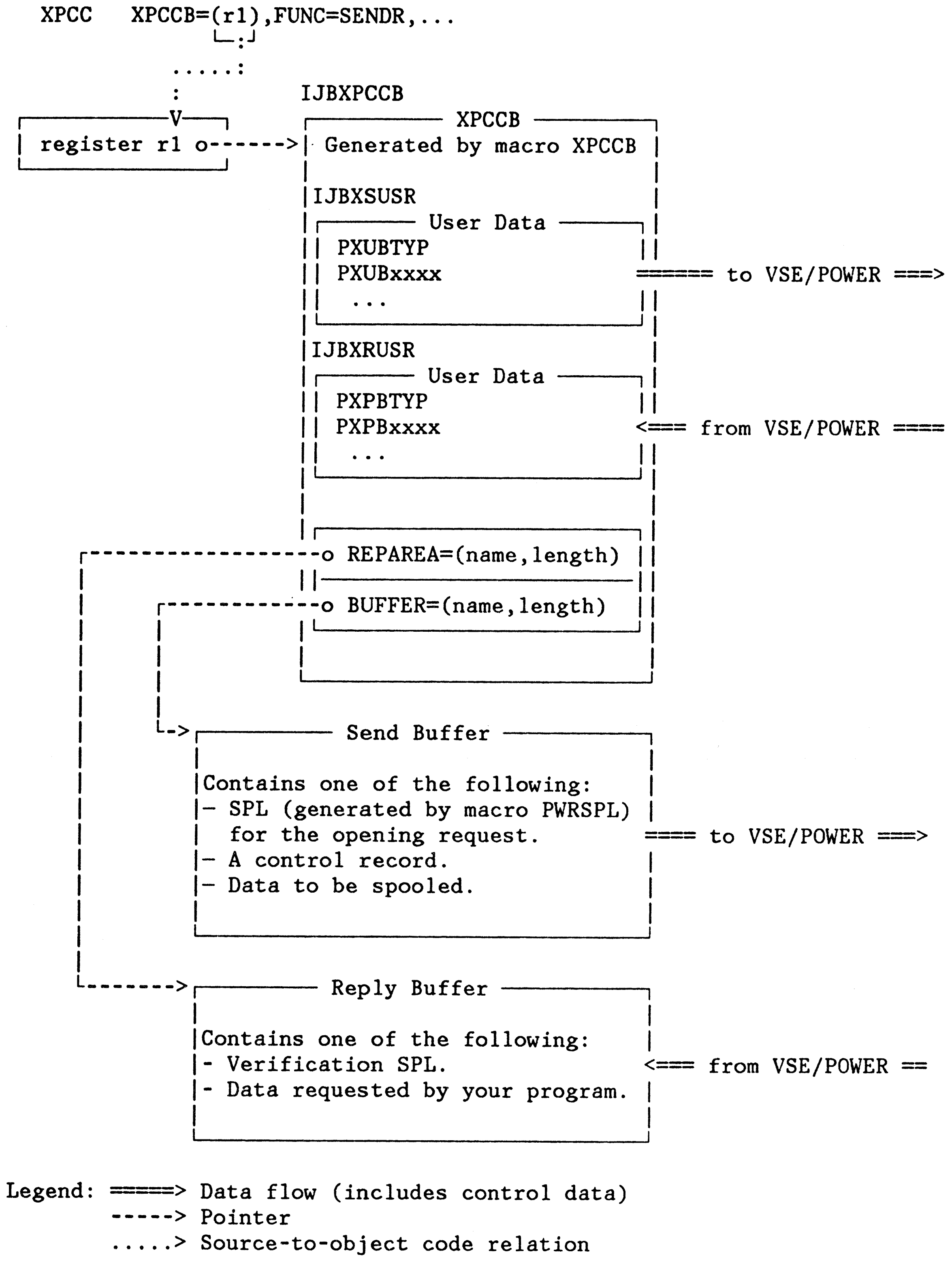


Figure 14. The Macros and Control Blocks for Spool Access

When your program issues a service request, the system passes the associated XPCCB and send buffer to VSE/POWER. The system returns to your program's XPCCB user data passed by VSE/POWER; it puts data into your program's reply buffer as applicable.

Before your program issues a request, it must ensure that the preceding request (if any) is complete.

Separate sections of this chapter deal with setting up a communication path, issuing access-service requests, and removing an existing communication path. In studying these sections, you may find it helpful to have an output listing for an assembly of the following macros:

```
MAPXPCCB
PWRSPL  TYPE=MAP
```

The assembler produced DSECTs include explanatory comments.

A complete list of the VSE/POWER return and feedback codes is given in the DSECT PXPUSER, which the assembler generates for a PWRSPL TYPE=MAP macro. You find the return codes at label PXPRETCD and the feedback codes at label PXPFBKCD.

To get a feel for the scope and level of the support, consider reading the description of the PWRSPL macro under "PWRSPL Macro" on page 99.

SET UP A COMMUNICATION PATH

Sequence of Coding: To set up a communication path between your program and VSE/POWER, include in your program coding in the sequence as shown in Figure 15 on page 34.

For all access-service requests via an existing path, your program must use the XPCCB which you supplied for program identification and connection. The section "Spool-Access Support Programming Example" on page 120 includes an identify and a connect coding sequence at labels IDENT and CONCT, respectively.

For each additional communication path established to VSE/POWER, the connection XPCCB control block must be copied from the original identification XPCCB. This is described under "Set Up Two or More Communication Paths" on page 97.

Return Information: Your program should test return information as follows:

1. Register 15.

2. The return code in the XPCCB field IJBXRETC.

   For a complete list of possible return codes, see the section "XPCC Macro" on page 113.

When the setup of the communication path is complete, your program can issue access-service requests.

```
    Coding in your
    application program                    Comments
    -------------------                    ------------------------------------

      ... ... ...
         |
         V
 XPCC  FUNC=IDENT,...                      Identifies your program to the
    Check the return codes in              system. (Required only once
    register 15 and in the                 per program.)
    XPCCB (byte IJBXRETC).
         |
         V
 XPCC  FUNC=CONNECT,...                    The macro must refer to the same
    Check the return codes in              XPCCB you used for program identi-
    register 15 and in the                 fication. (Required for each
    XPCCB (byte IJBXRETC).                 communication path to VSE/POWER.)
         |
         V
 WAIT  IJBXCECB                            Wait for the CONNECT ECB to be
         |                                 posted.
         V
      ... ... ...
```

Figure 15. Sequence Diagram — Setting up a Communication Path

REQUEST VSE/POWER ACCESS-SERVICES

You can access VSE/POWER for service requests as follows:

- CTL (control) service: one or more Requests to process a command that is being passed to VSE/POWER explicitly or by way of control values. This is discussed under "Request a CTL Service" on page 36.

- GET (retrieve spooled data) service: requests to retrieve a certain queue entry from the specified local VSE/POWER queue. For a discussion of this service, turn to "Request a GET Service" on page 41.

- PUT (submit job or output) service: requests to include, into the applicable VSE/POWER queue, a job (or job stream) or output data. For more information see the section "Request a PUT Service – General Considerations" on page 58.

Via an existing communication path, only one type of service processing can be handled at a time. You cannot, for example, open GET-service processing and issue a CTL-service request before the previously started GET processing is finished. For all requests which your program issues via the communication path, it must use the same XPCCB.

You define a request, and also control information needed by VSE/POWER, primarily in a PWRSPL-generated SPL; to some extent, you specify control information in the user area of the XPCCB or in a separate control record.

The requests for a desired service have to be coded in a certain sequence depending on the type of service. This sequence is shown in the form of a diagram followed by a discussion of the various requests.

Request a CTL Service

VSE/POWER can process only one control function per CTL-service request. Open a CTL-service request in your program if you want VSE/POWER to do one of the following:

- Pass a command for processing by VSE/POWER.
- Alter certain attributes of a queue entry.
- Cancel a job that is being executed.
- Delete a reader or an output queue entry.
- Display status information about a reader or an output queue entry or a group of entries.
- Release a job or an output queue entry.
- Place a reader or an output queue entry into the hold state.

Refer to Figure 16 on page 37, a coding sequence diagram. It shows the kind of coding you have to supply in your program and in what sequence this coding is to be. This coding is discussed in the subsequent paragraphs. The section "Spool-Access Support Programming Example" on page 120 includes a CTL-service request at label CTLA1.

Issue a Service-Open Request

To open this processing, VSE/POWER requires:

- Byte PXUBTYP of the XPCCB to be set to the value equated to PXUBTSPL. This tells VSE/POWER that the send buffer contains an SPL.

- In the send buffer, an SPL set up by a PWRSPL macro with TYPE=GEN or updated by a PWRSPL macro with TYPE=UPD.

- A reply buffer set up in your program either by specifying REPAREA=(areaname,length) or by inserting the buffer's address and length into the four-byte XPCCB fields IJBXRADR and IJBXRLNG, respectively. Any messages that VSE/POWER generates are returned to your program in this buffer (see also "Issue a Return-Message Request" below).

Processing for a CTL service may be discontinued at any time by either a quit request or by a new function request. This is discussed under "Issue a Service End Request" on page 38.

```
     Coding in your
     application program                      Comments
     -------------------         ------------------------------------
          |
          V
  Open the service
    XPCC  FUNC=SENDR,...         Your program's send buffer must
          |                      contain an SPL generated (or up-
          V                      dated) for processing a CTL service.
    Check the return codes in
    register 15 and in the
    XPCCB (byte IJBXRETC).
          |
          V
  WAIT  IJBXSECB                 Wait for the SENDR ECB to be posted.
          |                      This indicates that VSE/POWER has
          |                      finished processing the service.
    Check the reason code (in the
    XPCCB byte IJBXREAS)
          |
          V
    Check the VSE/POWER return and
    feedback codes (in the XPCCB
    bytes PXPRETCD and PXPFBKCD,
    respectively).
          |<-------------------------------+
          V                                |
    Check for and evaluate messages |      If messages are to be returned, then
    returned by VSE/POWER           |      VSE/POWER passes them to your pro-
          |                         |      gram's reply buffer.
          V                         |
  Get additional messages, if any   |      Coding for this purpose is required
    XPCC  FUNC=SENDR,...            |      only if the feedback code indicates
          |                         |      that more messages are queued. No
          |                         |      SPL need be passed for this request;
          |                         |      your program must set a request code
          V                         |      in the XPCCB.
  WAIT  IJBXSECB                    |      Wait for the SENDR ECB to be posted.
          |                         |
          V                         |
    Check the VSE/POWER return and  |
    feedback codes (this is the same|
    as above).  If more messages are|      Loop until VSE/POWER returns the
    waiting for transfer, then -----+      feedback code PXP00EOD.
    Else -+
          |
          V
  End of service
```

Figure 16. Sequence Diagram — CTL-Service Processing

Issue a Return-Message Request

VSE/POWER queues any messages that may occur while it processes the requested CTL service. It passes these messages to your program's reply buffer.

If all of the queued messages fit into your reply buffer, VSE/POWER indicates this by a return- and feedback-code combination PXPRCOK/PXP00EOD. If the generated messages do not fit, VSE/POWER passes to your program the return- and feedback-code combination PXPRCOK/PXP00OK.

To have VSE/POWER pass messages not yet transferred, your program must:

1. Set byte PXUACT1 of the XPCCB to the value equated to PXUATRMR.

2. Issue an XPCC FUNC=SENDR request which passes a null buffer (instead of an SPL).

The coding sequence at label DSPL2 in the section "Spool-Access Support Programming Example" on page 120 shows how to set up a null buffer and how to issue a return-message request.

VSE/POWER deletes messages queued but not yet transferred if your program does one of the following:

- Issues another, different service request.

- Issues a quit request.

- Ends communication via the currently used path.

Issue a Service End Request

If the processing of a CTL service is to be discontinued before it is finished, you can do either of the following:

- Issue a new function request, which requires an SPL to be passed to VSE/POWER.

- Issue a quit request. To do this:

  1. Set byte PXUBTYP of the XPCCB to zero.

  2. Set byte PXUACT1 of the XPCCB to the value equated to PXUATABR.

  3. Issue an XPCC FUNC=SENDR request passing a null send buffer, that is, a buffer with a length of zero.

  The coding sequence at label GQUIT in the section "Spool-Access Support Programming Example" on page 120 shows how to set up a null buffer and how to issue a quit request.

Check the Return Information

For the return information to be checked by your program after an XPCC request, refer to "XPCC Macro" on page 113.

Your program should also check the return information supplied by VSE/POWER. Provide for this checking after your program's SENDR ECB has been posted.

Figure 17 on page 40 lists the return and feedback codes that VSE/POWER may supply when it processes a CTL-service related request. The list is ordered in ascending order by code values. It relates the codes to the applicable request types and gives the names that are equated to the feedback codes. A complete list of the VSE/POWER return and feedback codes is given in the DSECT PXPUSER, which the assembler generates for a PWRSPL TYPE=MAP macro. You find the return codes at label PXPRETCD and the feedback codes at label PXPFBKCD.

			Request Type	
Mnemonic	Return Code	Feedback Code	CTL Open	Get Messages
PXP00OK	00	00	X	X
PXP00EOD		01	X	X
PXP04SOA	04	09	X	
PXP04DNF		0B	X	
PXP04TQN		0C	X	
PXP08SPL	08	01	X	
PXP08REQ		02	X	
PXP08SRQ		03	X	
PXP08FB2		04	X	
PXP08JNM		05	X	
PXP08QID		06	X	
PXP08CLS		07	X	
PXP08PWD		08	X	
PXP08UID		09	X	
PXP08BTS		1A	X	X
PXP08IAB		1C		X
PXP08CON		22	X	X
PXP08IBT		24	X	
PXP08BOS		27	X	
PXP08JSF		32	X	
PXP0CINS	0C	01	X	X
PXP0CIXF		02	X	X
PXP0CIOE		07	X	X
PXP10PSP	10	05	X	X
PXP10SIE		06	X	X

Figure 17. Return and Feedback Codes for CTL-Service Related Requests

Request a GET Service

You request a GET service if you want VSE/POWER to retrieve a certain queue entry and make this entry available to your program. In your program, you issue GET-service requests as follows:

1. An Open request to start the desired retrieval of spool data – For details, see "Issue a Service-Open Request" on page 44.

2. One or more GET spool data requests to have VSE/POWER make the desired spool data available to your program – For details, see "Issue a GET Spool Data Request" on page 47.

3. An end-service request, which may be one of the following:

   - A Close request to indicate that the retrieval of a specific queue entry is finished – For details, see "Issue a Close Request" on page 47.

   - A Quit request to end any further retrieval of spool data – For details, see "Issue a Quit Request" on page 48.

   - A Quit-and-Lock request to indicate, for example, that the processing of an output queue entry failed - For details, see "Issue a Quit-and-Lock Request" on page 48.

   - A Purge request to end any further retrieval of spool data and to Purge the accessed queue entry from its queue – For details, see "Issue a Purge Queue Entry Request" on page 48.

You may, in addition, issue:

- A Checkpoint request to record a suitable restart point should a restart be desirable or become necessary – For details, see "Request a Checkpoint" on page 49.

- A Restart request to set up the retrieval of a queue entry's spool data at a point other than the beginning – For details, see "Request a Restart" on page 50.

- A Get OPTB request to obtain one or more available output parameter text blocks (OPTBs) - For details, see "Issue a Get-OPTB Request" on page 52.

- A Modify OPTB request to change an OPTB - For details, see "Issue a Modify-OPTB Request" on page 53.

General Considerations

Accessing a Queue Entry: A queue entry, to be retrieved for your program, must have a disposition of D (dispatchable) or K (keep after processing). An exception is the retrieval of a queue entry in BROWSE mode; in other words, retrieval just for the purpose of examining (viewing on a screen, for example) the contents of the queue entry's data. In BROWSE mode, any queue entry can be retrieved, regardless of its disposition.

In addition, your program can get a queue entry only if one of the following is true:

- The program identifies itself as the owner of the job or output that is to be accessed.
- The queue entry is an output destined for your program.
- The queue entry is an output with a destination of ANY (which means that any program accessing VSE/POWER can get this output).

If a queue entry is password protected, your program must supply the matching password in the SPL.

Disposition of a Retrieved Queue Entry: If you end the retrieval of a queue entry by a Close request, then this retrieval is, for VSE/POWER, the same as processing this entry. Therefore, if the entry's disposition was

D VSE/POWER deletes the entry.
K VSE/POWER retains the entry with the entry's disposition changed to L.

For further information on disposition refer to the VSE/POWER Installation and Operations Guide.

Data Passed by VSE/POWER: If your program requests a RDR queue entry, VSE/POWER does not return the * $$ JOB, * $$ RDR, and * $$ EOJ statements.

Each record made available by VSE/POWER is preceded by an eight-byte prefix as shown in Figure 18 on page 43. A DSECT of this prefix, labeled RECPRFIX, is available to you if you issue a PWRSPL macro with TYPE=MAP.

Bytes	Explanation
0	Carriage control character or X'00'.
1	Record type: X'00' = A normal data record X'01' = A spool parameter list (SPL) X'03' = A separator-page (separator-card) start record X'04' = A 3540 data record (applies only to a RDR queue entry) X'05' = A control-command record (such as skip to channel 1 (X'8B) or block data check (X'73') X'06' = A CPDS (composed page data stream) record X'07' = A separator-page (separator-card) end record X'08' = An end-of-copy record
2-3	Length of a logical record (in binary notation)
4-7	VSE/POWER assigned record number (in binary notation); you can use this number to specify a restart point should a restart become necessary.

Figure 18. Record Prefix Layout

<u>The Verification SPL</u>: In response to your first (opening) request, VSE/POWER returns to your program a verification SPL. Consider analyzing this SPL in your program and coding programmed actions that may be necessary.

The verification SPL contains the same information as the SPL passed by your program. Some of the verification SPL's fields contain data about the currently accessed queue entry and not supplied by your program. Examples are: record format and length, number of print lines or pages. Your program may need this information for setting up output processing.

<u>Required Buffers</u>: Your program must provide buffers as follows:

- A send buffer for the opening request, large enough to hold the required SPL. You can define the buffer by way of the BUFFER operand of the XPCC or XPCCB macro.

- A reply buffer large enough to hold either of the following whichever is larger:

  - The verification SPL passed by VSE/POWER in response to your program's opening request.
  - The largest data record of the requested queue entry.
  - The largest OPTB.

You define the buffer by way of the REPAREA operand of the XPCCB macro.

Checkpoint and Restart Capability: Your program can request checkpoints to be recorded. Each time it records a checkpoint, VSE/POWER passes to your program a checkpoint-response record.

Your program can request VSE/POWER to restart the retrieval of spooled records, should the need arise. You can request this restart at a recorded checkpoint (normally the last) or at any other data record.

End the Service: Your program can end a GET-service at any time after completion of a relevant XPCC SENDR request. This is discussed further under "Issue an End-Service Request" on page 47.

Coding Sequence

Figure 19 on page 45, shows the kind and sequence of the coding needed in your program for the retrieval of a complete queue entry. This coding is discussed in the subsequent paragraphs. The section "Spool-Access Support Programming Example" on page 120 includes a GET-service request at label GETB1.

Issue a Service-Open Request

To open GET-service processing, VSE/POWER requires:

- Byte PXUBTYP of the XPCCB to be set to the value equated to PXUBTSPL. This indicates to VSE/POWER that the send buffer contains an SPL.
- An SPL as set up by a PWRSPL macro with TYPE=GEN or updated by a PWRSPL macro with TYPE=UPD.
- A reply buffer to which VSE/POWER passes the verification SPL.

```
    Coding in your
    application program                   Comments
    ___________________            _________________________________

        |
        V
 Open request
   XPCC  FUNC=SENDR,...            Your program's send buffer must
        |                          contain an SPL generated (or up-
        V                          dated) for processing a GET service.
   Check the return codes in
   register 15 and in the
   XPCCB (byte IJBXRETC).
        |
        V
 WAIT  IJBXSECB                    Wait for the SENDR ECB to be posted.
        |
        V
   Check the reason code (in the
   XPCCB byte IJBXREAS).
        |
        V
   Check the VSE/POWER return and
   feedback codes (in the XPCCB
   bytes PXPRETCD and PXPFBKCD,
   respectively).
        |
        V
   Check for and evaluate the SPL  VSE/POWER returns a verification SPL
   from VSE/POWER, if necessary.   to your program's reply buffer if
        |                          the request has been accepted and
        V                          can be processed by VSE/POWER.
   See next part
```

Figure 19 (Part 1 of 2). Sequence Diagram – GET Service for a Complete Queue Entry

```
     Coding in your
     application program                        Comments
     ___________________               ______________________________

         ┌── From preceding page
         |<─────────────────────┐
         V                      |
  GET spool data request        |
    XPCC  FUNC=SENDR,...        |   Your program's XPCCB must refer to
         |                      |   a zero-length send buffer.
         V                      |
    Check the return codes in re-|
    gister 15 and in the XPCCB  |
    (byte IJBXRETC).            |
         |                      |
         V                      |
  WAIT  IJBXSECB                |   Wait for the SENDR ECB to be posted.
         |                      |   VSE/POWER places the retrieved data
         V                      |   record(s) into your program's re-
    Check the return and feed-  |   ply buffer.
    back codes (this is the same|
    as in Part 1).              |
         |                      |
         V                      |
    Deblock the data in the reply|
    buffer, if necessary.  If more|  Loop until VSE/POWER returns the
    records are to be transferred ┘  feedback code PXPOOEOD.
    Else ┐
         |
         V
  End-retrieval request
    XPCC  FUNC=SENDR,...            Your program's XPCCB must refer to a
         |                          zero-length send buffer.
         V
    Check the return codes in
    register 15 and in the
    XPCCB (byte IJBXRETC).
         |
         V
  WAIT  IJBXSECB                    Wait for the SENDR ECB to be posted.
         |                          This ensures that the communication
         V                          is free for another service request.
    Check the VSE/POWER return and
    feedback codes (this is the
    same as in Part 1).
         |
         V
  End of Service
```

Figure 19 (Part 2 of 2). Sequence Diagram – GET Service for a Complete Queue Entry

Issue a GET Spool Data Request

After VSE/POWER has passed the verification SPL, your program will eventually issue one or more GET spool data requests, each one after the completion of the preceding other. You do this by code in your program for the following:

1. Set byte PXUBTYP of the XPCCB to zero.

2. Set byte PXUACT1 of the XPCCB to the value equated to PXUATSDR.

3. Issue an XPCC FUNC=SENDR request.

In response to a GET spool data request, VSE/POWER fills your program's reply buffer with records of the queue entry, one record behind the other. You define this buffer by the REPAREA operand of your XPCCB macro; you may want to alter this definition by changing the buffer's address (in field IJBXRADR) and its length (in field IJBXRLNG).

Issue an End-Service Request

When your program has finished processing the data of a queue entry, it should issue one of the following requests after VSE/POWER has completed a relevant XPCC SENDR request:

Close To have VSE/POWER dispose of the queue entry in accordance with VSE/POWER's disposition rules.

Quit To return the queue entry with its original disposition.

Quit-and-Lock
To indicate that the processing of an output queue entry failed.

Purge To purge the queue entry from the queue.

Issue a Close Request: In your program, you normally issue a Close request when VSE/POWER has completed the retrieval of the desired queue entry. However, you can issue a Close request any time during the retrieval of a queue entry.

When it receives a Close request, VSE/POWER handles the queue entry in accordance with its disposition rules. If the disposition is:

D VSE/POWER deletes the queue entry.
K VSE/POWER retains the queue entry with a disposition of L.

To issue a Close request in your program:

1. Set byte PXUBTYP of the XPCCB to zero.

2. Set byte PXUACT1 of your XPCCB to the value equated to PXUATRQS.

3. Issue an XPCC FUNC=SENDR request which passes a null buffer, that is, a buffer with a length of zero.

The coding in your program is similar to a quit request as shown at label GQUIT in the section "Spool-Access Support Programming Example" on page 120. However, the MVI instruction that sets byte PXUACT1 of the XPCCB is to be replaced by the sample instruction shown as comment with the label *GCLOSE.

Issue a Quit Request: You can do this at any point during the retrieval of a queue entry. The request causes VSE/POWER to retain the queue entry with its originally assigned priority and disposition.

To issue a quit request in your program:

1. Set byte PXUBTYP of the XPCCB to zero.

2. Set byte PXUACT1 of your XPCCB to the value equated to PXUATABR.

3. Issue an XPCC FUNC=SENDR request passing a null send buffer, that is, a buffer with a length of zero.

The coding sequence at label GQUIT in the section "Spool-Access Support Programming Example" on page 120 shows how to set up a null buffer and how to issue a quit request.

Issue a Quit-and-Lock Request: You can do this at any point during the retrieval of a queue entry. The request causes VSE/POWER to requeue the currently processed queue entry in the appropriate class chain with a temporary disposition of Y for the purpose of:

- Indicating that a problem has occurred during output processing, and

- Preventing that the output queue entry is handled again until the subsystem has taken some special action (for example, issued the PALTER command to alter the temporary disposition to a dispatchable one).

To issue a quit-and-lock request in your program:

1. Set byte PXUBTYP of the XPCCB to zero.

2. Set byte PXUACT1 of your XPCCB to the value equated to PXUAT1PF.

3. Issue an XPCC FUNC=SENDR request passing a null send buffer, that is, a buffer with a length of zero.

Issue a Purge Queue Entry Request: You can do this at any point during the retrieval of a queue entry. The request causes VSE/POWER to delete the currently processed queue entry from its queue.

To issue a Purge request in your program:

1. Set byte PXUBTYP of the XPCCB to zero.

2. Set byte PXUACT1 of your XPCCB to the value equated to PXUATPRG.

3. Issue an XPCC FUNC=SENDR request passing a null send buffer, that is, a buffer with a length of zero.

The coding in your program is similar to a quit request as shown at label GQUIT in the section "Spool-Access Support Programming Example" on page 120. However, the MVI instruction that sets byte PXUACT1 of the XPCCB is to be replaced by the sample instruction shown as comment with the label *GPURGE.

Request a Checkpoint

Checkpointing is meaningful if your program requests a large amount of spooled data to be retrieved. It is meaningful, for example, if your program is to process retrieved spool data in sections. It can save processing time should a program or system failure occur.

Your program can request VSE/POWER to record a checkpoint at any time between two GET spool data requests. VSE/POWER records checkpoint information as follows:

- Logical record number as specified in the checkpoint-control record.
- The output-copy number (if applicable).

To have VSE/POWER record a checkpoint, your program must:

1. Set up a checkpoint-control record in your program's send buffer.

   By issuing a PWRSPL macro with TYPE=MAP, the assembler generates a DSECT of this record at label PXCPDSCT.

   In the checkpoint-control record, you can specify a copy number (field PXCPRCPY) if the control record applies to an output queue entry. The number tells VSE/POWER, that checkpoint information is to be recorded for the specified record in the specified output copy. If you set the field to zero, VSE/POWER uses its current number-of-copies count.

2. Set byte PXUACT1 of the XPCCB to zero.

3. Set byte PXUBTYP of the XPCCB to the value equated to PXUBTCTL.

   This tells VSE/POWER that your program's send buffer contains a control record.

4. Issue an XPCC FUNC=SENDR request.

   The request passes to VSE/POWER the checkpoint-control record which your program has set up in its send buffer.

After having recorded the requested checkpoint, VSE/POWER returns a checkpoint-response record (in your program's reply buffer). The assembler generates a DSECT of this record at label PXCRDSCT if you issue a PWRSPL macro with TYPE=MAP.

As described in the section "Request a Restart," VSE/POWER returns the last recorded checkpoint of a queue entry when a retrieval of this queue entry is opened again. In your program, you can then decide whether VSE/POWER is to continue retrieval at that checkpoint (by issuing a restart request) or from the beginning (by issuing a GET spool data request).

Request a Restart

Your program can request VSE/POWER to restart retrieval at any point during GET data processing. It can request such a restart immediately after processing of the Open request is complete; it can, in fact, request a restart even after the end-of-data indication has occurred, but before it passes the end-service request.

Figure 20 on page 51 shows a sequence diagram for a restart request. The diagram assumes that GET service processing has been opened successfully. The section "Spool-Access Support Programming Example" on page 120 includes a restart request at label GETB3.

```
   Coding in your
   application program                              Comments
   -------------------                     ----------------------------------
       |
       V
WAIT  IJBXSECB                             Wait for the SENDR ECB to be posted.
       |                                   VSE/POWER returns a verification
       V                                   SPL to your program's reply buffer.
  Check the reason code (in the
  XPCCB byte IJBXREAS).
       |
       V
  Pick up and evaluate the veri-
  fication SPL, if necessary.
       |
       V
Restart request
  XPCC  FUNC=SENDR,...                     Your program's send buffer must
       |                                   contain a restart control record.
       V
  Check the return codes in
  register 15 and in the
  XPCCB (byte IJBXRETC).
       |
       V
WAIT  IJBXSECB                             Wait for the SENDR ECB to be posted.
       |                                   VSE/POWER transfers data records to
       V                                   your program's reply buffer, as many
  Check the VSE/POWER return and           records as will fit.
  feedback codes (in the XPCCB
  bytes PXPRETCD and PXPFBKCD,
  respectively).                           At this point, the coding sequence
       |                                   is the same as for the retrieval of
       V                                   a complete queue entry.
```

Figure 20. Sequence Diagram – Restart of a GET Service

To make a restart request, your program must:

1. Set byte PXUACT1 of the XPCCB to zero.

2. Set byte PXUBTYP of the XPCCB to the value equated to PXUBTCTL.

   This tells VSE/POWER that your program's send buffer contains a control record.

3. Set up a restart control record in your program's send buffer.

   By issuing a PWRSPL macro with TYPE=MAP, the assembler generates the restart-control record DSECT labeled PXRSDSCT. In the record, set field PXRSOPT to:

X'00' if the number in field PXRSRECN is a spool-record (card) number.
X'20' if the number in field PXRSRECN is a page number.
X'80' if the number in field PXRSRECN is a line number.

If an output queue entry is being retrieved, you can specify a copy number in field PXRSCOPN of the control record. The number tells VSE/POWER that it is to restart retrieval at the specified record in the specified output copy. If you set the field to zero, VSE/POWER uses the current number-of-copies count.

As a help in defining a restart point, VSE/POWER passes to your program the internal record count in field RECLOGNO of each retrieved record's prefix. If your program accesses a previously retrieved and checkpointed queue entry, VSE/POWER gives you the last recorded checkpoint information in the verification SPL as follows:

- The number of the logical record last checkpointed – In field SPLDCLC.
- The related copy number, if applicable – In field SPLDCCPY.

4. Issue an XPCC FUNC=SENDR request.

   The request passes to VSE/POWER the restart-control record set up by your program in its send buffer.

In response to a valid restart request, VSE/POWER repositions the retrieval pointer. VSE/POWER then continues processing by passing records to your reply buffer, starting with the record or line defined in the restart control record.

Issue a Get-OPTB Request

Your program can request VSE/POWER to retrieve either all available OPTBs (output parameter text block) or a specific OPTB.

- OPTBs are contained in an output queue entry if the * $$ LST or * $$ PUN statement includes any user-defined keywords that have been defined in autostart DEFINE statements.

- OPTBs can also be passed to VSE/POWER as an appendage of the SPL (Spool Parameter List) at PUT Open time (see "Specify Output Parameter Text Blocks (OPTBs)" on page 91).

You can send the Get-OPTB control record to VSE/POWER at any time while accessing an output queue entry (during GET data processing) or while spooling output data (PUT function). If OPTBs are present, the SPL contains a two-byte field indicating the total length of all OPTBs (see Figure 35 on page 92 and Figure 36 on page 92).

You can obtain the format of the GET-OPTB control record by issuing a PWRSPL macro with TYPE=MAP. The assembler generates the GET-OPTB control record DSECT labeled PXGODSCT. In the control record, pass the desired OPTB identifier in field PXGOID.

If you specify an OPTB identifier in the control record, VSE/POWER places only this particular OPTB into your program's reply buffer. If you do not specify an OPTB ID, VSE/POWER places all OPTBs into the reply area.

To obtain one or more OPTBs your program must:

1. Set up a Get-OPTB control record in your program's send buffer.

2. Set byte PXUACT1 of the XPCCB to zero.

3. Set byte PXUBTYP of the XPCCB to the value equated to PXUBTCTL.

   This tells VSE/POWER that your program's send buffer contains a control record.

4. Issue an XPCC FUNC=SENDR request.

   The request passes to VSE/POWER the Get-OPTB control record set up by your program in its send buffer.

Issue a Modify-OPTB Request

Your program can request VSE/POWER to modify an existing OPTB. Via the Modify-OPTB control record you can update (overwrite) any OPTB with a new one, which must have the same length as the old OPTB. You can send the Modify-OPTB control record to VSE/POWER at any time while accessing an output queue entry (during GET data processing) or while spooling output data (PUT function), but not when you are in browse mode.

You can obtain the format of the Modify-OPTB control record by issuing a PWRSPL macro with TYPE=MAP. The assembler generates the Modify-OPTB control record DSECT labeled PXMODSCT. In the control record, pass the OPTB to be modified starting at field PXMOOPTB.

To modify one or more OPTBs your program must:

1. Set up a Modify-OPTB control record in your program's send buffer.

2. Set byte PXUACT1 of the XPCCB to zero.

3. Set byte PXUBTYP of the XPCCB to the value equated to PXUBTCTL.

   This tells VSE/POWER that your program's send buffer contains a control record.

4. Issue an XPCC FUNC=SENDR request.

   The request passes to VSE/POWER the Modify-OPTB control record set up by your program in its send buffer.

Check the Return Information

For the return information to be checked by your program after an XPCC request, refer to "XPCC Macro" on page 113.

For each individual GET-service request, your program should check return information supplied by VSE/POWER. Provide for this checking after your program's SENDR ECB has been posted.

Figure 21 on page 55 lists the return and feedback codes that VSE/POWER may supply when it processes a GET-service related request. The list is ordered in ascending order by code values; it relates the codes to the applicable request types; it gives the names that are equated to the feedback codes.

A complete list of the VSE/POWER return and feedback codes is given in the DSECT PXPUSER, which the assembler generates for a PWRSPL TYPE=MAP macro. You find the return codes at label PXPRETCD and the feedback codes at label PXPFBKCD.

			Request Type						
Mnemonic	Return Code	Feedback Code	GET Open	BROWSE Open	GET Data	Check-point	Re-start	GET OPTB	Modify OPTB
PXP00OK	00	00	X	X	X	X	X	X	X
PXP00EOD		01			X		X		
PXP04NOF	04	01	X	X					
PXP04JOP		02	X	X					
PXP04BSY		03	X	X					
PXP04NDS		04	X						
PXP04RER		06					X		
PXP04CER		07				X			
PXP04SOA		09	X	X					
PXP04BER		0A							X
PXP04ONF		11						X	X
PXP08SPL	08	01	X	X					
PXP08REQ		02	X	X					
PXP08JNM		05	X	X					
PXP08QID		06	X	X					
PXP08CLS		07	X	X					
PXP08PWD		08	X	X					
PXP08UID		09	X	X					
PXP08BTS		1A			X	X	X	X	
PXP08IAB		1C			X				
PXP08ICR		1D				X	X	X	X
PXP08CON		22	X	X	X	X	X		
PXP08IBT		24	X	X		X	X		
PXP08ROS		25			X				
PXP08SOS		26	X	X					
PXP08BOS		27				X	X		
PXP08FB1		2B	X	X					
PXP08JNO		31	X	X					
PXP08JSF		32	X	X					
PXP08IRR		38						X	X
PXP08IOP		39							X
PXP08OLM		3A							X
PXP08IDH		3D						X	X
PXP0CINS	0C	01	X	X	X	X	X		
PXP0CIXF		02	X	X	X	X	X		
PXP0CIOE		07	X	X	X	X	X		
PXP10PSP	10	05	X	X	X	X	X		
PXP10SIE		06	X	X	X	X	X		

Figure 21 (Part 1 of 2). Return and Feedback Codes for GET-Service Requests

			Request Type				
Mnemonic	Return Code	Feedback Code	Purge	Close	Quit	Quit Lock	Flush Hold*
PXP00OK	00	00	X	X	X	X	X
PXP00EOD		01					
PXP04NOF	04	01					
PXP04JOP		02					
PXP04BSY		03					
PXP04NDS		04					
PXP04RER		06					
PXP04CER		07					
PXP04SOA		09					
PXP04BER		0A	X	X	X	X	X
PXP08SPL	08	01					
PXP08REQ		02					
PXP08JNM		05					
PXP08QID		06					
PXP08CLS		07					
PXP08PWD		08					
PXP08UID		09					
PXP08BTS		1A					
PXP08IAB		1C	X	X	X	X	X
PXP08ICR		1D					
PXP08CON		22	X	X	X	X	X
PXP08IBT		24					
PXP08ROS		25	X	X	X	X	X
PXP08SOS		26					
PXP08BOS		27					
PXP08RPH		28					X
PXP08FB1		2B					
PXP08JSF		32					
PXP0CINS	0C	01	X	X	X	X	X
PXP0CIXF		02	X	X	X	X	X
PXP0CIOE		07	X	X	X	X	X
PXP10PSP	10	05	X	X	X	X	X
PXP10SIE		06	X	X	X	X	X

* The flush hold function is part of the external device support.

Figure 21 (Part 2 of 2). Return and Feedback Codes for GET-Service Requests

Abnormal-End Condition During GET

- For an abnormal end of your program or of the VSE/POWER service task:

  If the output being retrieved by the GET service has been created via the PUT service with the 'protect' option on, the queue entry is placed into the non-dispatchable queue with disposition Y.

  A queue entry is protected when the SPL field SPLDMOHP is set on to signal: 'Hold when print/punch fails' (see Figure 26 on page 74).

- For an abnormal end of VSE/POWER or of the VSE system, the same applies as described above.

A queue entry with disposition Y is not automatically processed by the various VSE/POWER tasks. Your program can make use of the CTL service to

- Get a display of all queue entries that have a disposition of Y by entering the PDISPLAY ALL,CDISP=Y command, and

- Alter this disposition for a queue entry to make it eligible for processing again. To reset disposition Y of a queue entry to its original one, use the PALTER queue,jobname,DISP=* command.

For further information on disposition refer to the VSE/POWER Installation and Operations Guide.

Request a PUT Service – General Considerations

Your program would initiate PUT-service processing whenever data is to be submitted to VSE/POWER for inclusion in one of its queues. Jobs, including the associated input data, are submitted for inclusion in the RDR or XMT queue, whichever applies. Output data is submitted for inclusion in an output queue (LST, PUN, or XMT).

Submission for Inclusion in the XMT Queue

To submit a job for processing at another node (of your computer system's network), specify this in the * $$ JOB statement for the job.

To submit output data for transmission to another node, give the target node's name and the applicable user-ID in the SPL fields SPLDTNN and SPLDTUID, respectively.

In the SPL macro, you specify QUEUE=RDR for job input; you specify QUEUE=LST for list output and QUEUE=PUN for punch output.

Data Format

The format of the data to be spooled is always the same. Each record must be preceded by an eight-byte prefix as shown below (a DSECT of this prefix, labeled RECPRFIX, is available to you if you issue a PWRSPL macro with TYPE=MAP):

Bytes	Explanation
0	Carriage control character, if any.
1	Record type: X'00' = A normal data record. X'06' = A CPDS (composed page data stream) record.
2-3	Length of a logical record (in binary notation).
4-7	Reserved.

Data Lengths

For normal (type X'00') records, the minimum and maximum lengths are:

	Job Data	Output Data
Minimum	80 see *	1
Maximum	128	32K-1

* The default assumed by VSE/POWER if your program does not define a data length (in field SPLDLREC of the SPL).

If an output-spool record includes trailing blanks, your program can truncate these blanks prior to passing the record to VSE/POWER. This makes better use of buffer space.

If a record to be passed is longer than the specified maximum length, VSE/POWER truncates the record and informs your program by a feedback code; VSE/POWER spools the truncated record as well as the remaining records in the passed data buffer. For a passed record shorter than the specified maximum length, VSE/POWER:

- Expands this record by padding it with blanks at the end if a job is being submitted.
- Spools the record as presented if output is being submitted.

Size of Buffers

Your program must define the sizes of your send and reply buffers.

The Send Buffer: The buffer must be large enough to hold your program's SPL when the processing of the desired service is initiated. It must be large enough to hold the longest record (including the eight-byte prefix) that is to be passed to VSE/POWER.

To pass data to VSE/POWER for spooling, your buffer should have a length equal to the sum of the lengths of the data records (including the record prefix) that your program is to submit at a time. This may be just one record or a number of records. A zero data length field in a record prefix is an end-of-buffer indication for VSE/POWER.

You use the BUFFER operand of the XPCC macro or the XPCCB macro to define the buffer.

The Reply Buffer: The buffer must be large enough to hold a verification SPL passed to your program by VSE/POWER. You use the REPAREA operand of the XPCCB macro to define the buffer.

Retrieval of Messages

VSE/POWER collects all job- or output-specific messages that would normally go to the system console. They enable your program to determine whether the job- or output-spool operation was completed successfully; they inform your program about possible errors and unusual conditions, if any.

Following your Close request, VSE/POWER sets byte PXPINFO of your program's XPCCB to the value equated to PXPIMSG if any messages have been queued. Your program can request these messages to be passed by VSE/POWER.

If your program does not request the messages to be returned, VSE/POWER discards them on receipt of the next service request (CTL, GET, or PUT specified in FUNC=code of the PWRSPL macro).

Messages returned by VSE/POWER have a maximum length of 132 bytes; they are preceded by an eight-byte header. This header has a format as follows:

Bytes	Contents/Explanation
0	X'00' Set by VSE/POWER
1	X'02' Set by VSE/POWER
2-3	Length of message (in binary)
4-7	Reserved

A reply buffer of 700 bytes, for example, can hold up to five messages of maximum length.

Request a PUT Service - Submission of Jobs or Job Streams

This PUT service spools the submitted records as a queue entry in the RDR (XMT) queue.

In your program, you can issue job-related PUT-service requests as follows:

- An Open request to start the spooling of one or more jobs - For details, see "Issue a Service-Open Request" on page 62.
- One or more PUT spool data requests to have VSE/POWER spool the submitted job(s) - For details, see "Issue a PUT Spool Data Request" on page 65.
- A Close request to indicate that the submission of job data is finished and that the submitted job data is to be included in

VSE/POWER's input queues - For details, see "Issue a Close-Service Request" on page 66.

- A quit request to indicate that no further data is to be submitted for the currently processed job and that the job should not be included in VSE/POWER's input queues - For details, see "Issue a Quit Request" on page 66.

When your program submits job records, VSE/POWER does not insert any JECL statements. In other words, JECL statements required by VSE/POWER are to be supplied by your program preceding the job records.

If there is a user-written RDREXIT routine for local input, VSE/POWER passes to the routine the VSE job-control and JECL statements of the submitted jobs.

With one PUT-service request, your program can submit just one job or a job stream consisting of two or more jobs. However, submission of just one job per service request is the preferred method; it makes evaluation of returned messages easier. VSE/POWER does not return messages to your program until the end of data has been reached; as a result, a clear distinction as to which message belongs to which job is difficult if you submit two or more jobs following a PUT-service Open request.

After having processed one of the following, VSE/POWER returns to your program a verification SPL:

- Your Open request.

- Your Close request.

- A PUT-data request for a buffer containing two or more jobs if a short-on-account-space error occurs.

Besides the data supplied by your program in its SPL, a verification SPL contains:

- Default values for fields not set in your program's SPL.

- Statistics such as the total number of records spooled for your job if the verification SPL is passed by VSE/POWER following a Close request.

Coding Sequence

Refer to Figure 22 on page 63, a coding sequence diagram for the submission of a job to an input queue. Figure 22 shows the kind of coding you have to supply in your program and in what sequence this coding is to be. The section "Spool-Access Support Programming Example" includes a PUT-job service request at label PUTA1.

Issue a Service-Open Request

To open a PUT service for the submission of a job, VSE/POWER requires:

- Byte PXUBTYP of the XPCCB to be set to the value equated to PXUBTSPL. This indicates to VSE/POWER that the send buffer contains an SPL.

- An SPL as set up by a PWRSPL macro with TYPE=GEN or updated by a PWRSPL macro with TYPE=UPD.

  VSE/POWER requires control data to be passed in your program's SPL in addition to that specified in the JECL statements for the job. Figure 23 on page 65 lists the applicable SPL fields. For the lengths and data types of these fields, see the SPL DSECT which you get by issuing a PWRSPL macro with TYPE=MAP. Examine the fields of the SPL DSECT (at label SPLDS) and decide which of the SPL fields your program should set or change prior to the request.

- A reply buffer to which VSE/POWER passes the verification SPL.

```
   Coding in your
   application program                            Comments
   ___________________                  ________________________________

         |
         V
Open the request
  XPCC  FUNC=SENDR,...                  Your program's send buffer must
         |                              contain an SPL generated (or up-
         V                              dated) for processing a PUT-job
  Check the return codes in             service request.
  register 15 and in the
  XPCCB (byte IJBXRETC).
         |
         V
WAIT  IJBXSECB                          Wait for the SENDR ECB to be posted.
         |                              VSE/POWER passes a verification SPL
         V                              to your program's reply buffer.
  Check the reason code (in the
  XPCCB byte IJBXREAS).
         |
         V
  Check the VSE/POWER return and
  feedback codes (in the XPCCB
  bytes PXPRETCD and PXPFBKCD,
  respectively).
         |
         V
  Pick up and evaluate the veri-
  fication SPL, if necessary.
         |
         V
  See next part
```

Figure 22 (Part 1 of 2). Sequence Diagram — PUT Service, Job Submission

```
    Coding in your
    application program                       Comments
    ___________________                ____________________________

       ┌── From Part 1
       |<───────────────────────┐
       V                        |
 PUT-Data Request               |
   XPCC  FUNC=SENDR,...         |  Your program's send buffer must
       |                        |  contain the records which VSE/POWER
       V                        |  is to spool.
   Check the return codes in re-|
   gister 15 and in the XPCCB   |
   (byte IJBXRETC).             |
       |                        |
       V                        |
 WAIT  IJBXSECB                 |  Wait for the SENDR ECB to be posted.
       |                        |  It indicates to your program that
       V                        |  VSE/POWER has finished processing
   Check the VSE/POWER return   |  the records in the send buffer.
    and feedback codes (this is |
    the same as in Part 1).     |
   Either                       |
    Fill your buffer with records| Loop until all records for the
    for the next request and ───┘  queue entry have been passed.
   Or ──┐
       V
 Close request
   XPCC  FUNC=SENDR,...            Your program can make this request
       |                           with data in its send buffer or
       V                           with a null buffer being passed to
   Check the return codes in       VSE/POWER.
   register 15 and in the
   XPCCB (byte IJBXRETC).
       |
       V
 WAIT  IJBXSECB                    Wait for the SENDR ECB to be posted.
       |                           VSE/POWER passes a verification SPL
       V                           to your program's reply buffer.
   Check the reason code (in the
    XPCCB byte IJBXREAS).
   Check the VSE/POWER return and
    feedback codes (this is the
    same as in Part 1).
   Pick up and evaluate the veri-
    fication SPL, if necessary.
       |
       V
 End of service
```

Figure 22 (Part 2 of 2). Sequence Diagram - PUT Service, Job Submission

Legend: M = Mandatory; O = Optional

Name of Field		Purpose/Contents
SPLGQI	M	Queue identifier[1]
SPLGUS	M	User identifier[1]
SPLDPRGN	O	Programmer name[2]
SPLDROOM	O	Room number[2]
SPLDDEPT	O	Department number[2]
SPLDBLDG	O	Building number[2]
SPLDLREC	O	Maximum record length

[1] Normally defined in the PWRSPL macro along with other spool-control values.

[2] A * $ $ JOB specification overrides these operands.

You may have to supply additional spool-control values for the request (for example job name) by way of an * $$ JOB statement. Submit this statement as the first one of a job.

Figure 23. SPL Fields Applicable to a PUT-Job Service Request

Issue a PUT Spool Data Request

After VSE/POWER has passed the verification SPL, your program must issue one or more spool-data requests, each one after the completion of the preceding other. To do this, provide that your program:

1. Fills its buffer with records that are to be spooled by VSE/POWER.

2. Sets byte PXUBTYP of the XPCCB to the value equated to PXUBTNDB. This indicates that your program's send buffer contains records which are to be spooled.

3. Issues an XPCC FUNC=SENDR request.

VSE/POWER spools the records contained in your send buffer, except when an error condition is encountered. VSE/POWER indicates successful completion (or error, if any) to your program by way of return and feedback codes.

Issue a Close-Service Request

A Close request causes the data submitted up to this point to be placed into the RDR (XMIT) queue as a complete queue entry.

In your program, you can issue a Close request either:

1. Together with passing the last buffer of spool records for the queue entry being submitted, or

2. Separately after your program has passed this last buffer.

For either case, set byte PXUACT1 of the XPCCB to the value equated to PXUATEOD before you issue the requesting XPCC macro. For case 1, this is all you have to do.

For case 2, a separate Close request following the transfer of the last buffer, VSE/POWER requires that your program:

1. Sets byte PXUBTYP of the XPCCB to zero.

2. Sets up a null send buffer (by setting field IJBXBLN to the value equated to IJBXBLN).

3. Issues an XPCC FUNC=SENDR request.

The coding sequence at label PUTA3 in section "Spool-Access Support Programming Example" at the end of this chapter shows how to issue a Close request together with the last buffer of data records.

When it receives a Close request, VSE/POWER expects the last record in the last buffer of spool records to be a valid job-end statement. If a valid job-end statement is not supplied, VSE/POWER automatically adds this statement and queues a message about this for your program.

When all records of your job are queued, VSE/POWER returns a verification SPL to your program's reply buffer. This SPL contains descriptive job information such as VSE/POWER assigned default values and the job number. However, if your program submits two or more jobs before it passes a Close request, then the verification SPL reflects the characteristics of only the last job.

Issue a Quit Request

In your program, you may have to provide for a quit-type end of service processing; that is, end of the opened processing without any data to be queued by VSE/POWER.

Your program can issue a quit request any time after an individual PUT-service request is complete (which is indicated by a posting of field IJBXSECB of the XPCCB). A quit request causes VSE/POWER to purge the queue entry that is being built. In case of a multi-job submission, a job previously queued by VSE/POWER during the same PUT service remains unaffected.

Your program should check the quit-request return and feedback codes for successful completion of the request. This ensures that the communication path to VSE/POWER is free again for opening another service request.

If additional jobs are to be submitted for spooling, your program must reopen the PUT service by issuing an XPCC macro that passes a suitable SPL.

To issue a quit request in your program:

1. Set byte PXUBTYP of the XPCCB to zero.

2. Set byte PXUACT1 of your XPCCB to the value equated to PXUATABR.

3. Issue an XPCC FUNC=SENDR request passing a null send buffer, that is, a buffer with a length of zero.

This is the same as a quit request for a GET service. The section "Spool-Access Support Programming Example" on page 120 includes a coding sequence for a quit request at label GQUIT.

Issue a Return-Message Request

A general discussion of message retrieval is given under "Retrieval of Messages" on page 60.

VSE/POWER makes the generated messages available on request. Your program can pick them up in the defined reply buffer, one message behind the other.

If all messages fit into the reply buffer, VSE/POWER indicates this by the return- and feedback-code combination PXPRCOK and PXP00EOD. If additional messages are waiting to be transferred, VSE/POWER passes to your program a return- and feedback-code combination of PXPRCOK and PXP000K. In that case, your program should issue another return-message request.

VSE/POWER deletes messages queued but not yet transmitted if your program does one of the following:

- Issues another, different open-service request passing a new SPL.

- Issues a quit request.

- Ends communication via the currently used path.

Figure 24 on page 69, a coding sequence diagram, shows the kind of coding you have to supply in your program and in what sequence this coding is to be. Figure 24 assumes that PUT-data requests have been serviced by VSE/POWER for the complete queue entry.

The Section "Spool-Access Support Programming Example" on page 120 includes a message-retrieval request at label PUTA4. This coding sequence gets control if VSE/POWER passed XPCCB-user data with byte PXPINFO containing the value equated to PXPIMSG.

```
     Coding in your
     application program                            Comments
     -------------------                 ------------------------------------

          |
          V
  Close request
    XPCC  FUNC=SENDR,...                 Your program issues a Close request
          |                              when all records of a job have been
          V                              submitted.
    Check the return codes in
    register 15 and in the
    XPCCB (byte IJBXRETC).
          |
          V
  WAIT  IJBXSECB                         The WAIT required in your program to
          |                              ensure that VSE/POWER has finished
          |                              the necessary Close processing.
    Check the reason code in the         VSE/POWER returns a verification
     XPCCB.                              SPL.
    Check the VSE/POWER return and
     feedback codes.
    Pick up and evaluate the veri-
     fication SPL, if necessary.
|   Check info byte for queued
     messages.
          |<-------------------------+
          V                          |
  Return-message request             |   No SPL need be transferred for this
    XPCC  FUNC=SENDR,...             |   request; your program must set a re-
          |                          |   quest code in the XPCCB.
          V                          |
  WAIT  IJBXSECB                     |   Wait for the SENDR ECB to be posted.
          |                          |
          V                          |
    Check the VSE/POWER return and   |
     feedback codes (this is the     |   No SPL is required for this request;
     same as above).  If more mes-   |   your program must set a request code
     sages are to be transferred by  |   Loop until VSE/POWER returns the
     VSE/POWER, then ----------------+   feedback code PXPOOEOD.
    Else -+
          |
          V
    End of Service
```

Figure 24. Sequence Diagram — Retrieve Messages After a PUT-Job Service

To have VSE/POWER pass messages, your program must:

1. Set byte PXUBTYP of the XPCCB to zero.
2. Set byte PXUACT1 of the XPCCB to the value equated to PXUATRMR.
3. Issue an XPCC FUNC=SENDR request passing a null send buffer, that is, a buffer with a length of zero. The coding sequence at label PUTA4 in the section "Spool-Access Support Programming Example" on page 120 shows how to set up a null buffer.

Check the Return Information

For the return information to be checked by your program after an XPCC request, refer to "XPCC Macro" on page 113.

For each individual PUT-job type service request, your program should also check the return information supplied by VSE/POWER. Provide for this checking after your program's SENDR ECB has been posted.

Figure 25 on page 71 lists the return and feedback codes that VSE/POWER may supply when it processes a PUT-service related request for job submission. The list is in ascending order by code values. It relates the codes to the applicable request types and gives the names that are equated to the feedback codes.

A complete list of the VSE/POWER return and feedback codes is given in the DSECT PXPUSER, which the assembler generates for a PWRSPL TYPE=MAP macro. You find the return codes at label PXPRETCD and the feedback codes at label PXPFBKCD.

			Request Type				
Mnemonic	Return Code	Feedback Code	PUT Open	PUT Data	Close	Quit	Get Message
PXP00OK	00	00	X	X	X	X	X
PXP00EOD		01					X
PXP00NJB		02			X		
PXP00NRS		03			X	X	
PXP00RTR		04		X	X		
PXP00ZBF		05		X			
PXP04SOD	04	08		X	X		
PXP04SOA		09	X				
PXP08SPL	08	01	X				
PXP08REQ		02	X				
PXP08QID		06	X				
PXP08UID		09	X				
PXP08BTS		1A					X
PXP08IAO		1B	X				
PXP08IAB		1C		X			
PXP08PRG		1E	X				
PXP08ROO		1F	X				
PXP08DPT		20	X				
PXP08BLD		21	X				
PXP08CON		22	X	X	X	X	X
PXP08ROL		23		X			
PXP08IBT		24	X	X			
PXP08ROS		25			X	X	
PXP08SOS		26	X	X	X		
PXP08BOS		27	X				
PXP08RPH		28	X				
PXP08RPW		2A		X			
PXP08FB1		2B	X				
PXP08IML		2C	X				
PXP08SPA		2E		X			
PXP0CINS	0C	01	X	X	X	X	X
PXP0CIXF		02	X	X	X	X	X
PXP0CBTL		03		X			
PXP0CIOE		07	X	X	X	X	X
PXP10PSP	10	05	X	X	X	X	X
PXP10SIE		06	X	X	X	X	X

Figure 25. Return and Feedback Codes for PUT-Job Service Requests

Request a PUT Service - Submission of Output Data

This PUT service spools the submitted records as a queue entry in an output (LST, PUN, or XMT) queue. In your program, you issue output-related PUT-service requests as follows:

- An Open request to start the spooling of output:
  - To create a new output queue entry. This is the same as for the opening of a job-related PUT service; for details see "Issue a Service-Open Request" on page 62.
  - To restart an existing queue entry. This is discussed under "Request a Restart" on page 86.
  - To append output to an existing queue entry. This is discussed under "Append Output to an Existing Spool File" on page 89.
  - To specify Output Parameter Text Blocks (OPTBs). This is discussed under "Specify Output Parameter Text Blocks (OPTBs)" on page 91.
- One or more PUT spool data requests to have VSE/POWER spool the submitted output. This is the same as for the submission of job-related spool data; for details, see "Issue a PUT Spool Data Request" on page 65.
- A Close request to end the submission of output:
  - If there is no need to add additional spool data later on - This is the same as for the closing of a job-related PUT service; for details, see "Issue a Close-Service Request" on page 66.
  - If additional spool data is to be added later on - This is discussed under "Append Output to an Existing Spool File" on page 89.
- A quit request to indicate that no further data is to be submitted and that the output so far spooled is not to be included in a VSE/POWER output queue. This is the same as for a quit request during the spooling of job-related data; for details, see "Issue a Quit Request" on page 66.
- An output-segmentation request. For details, see "Request Output-Segmentation" on page 82.
- A checkpoint request. For details, see "Request a Checkpoint" on page 84.
- A restart request. For details, see "Request a Restart" on page 86.
- A Get-OPTB request. This is the same as for a Get-OPTB request during Get service processing; for details see "Issue a Get-OPTB Request" on page 52.

- A Modify-OPTB request. This is the same as for a Modify-OPTB request during GET service processing; for details see "Issue a Modify-OPTB Request" on page 53.

There is one major difference between this service processing and the submission of a job: for the spooling of output, a number of the fields of the required SPL may have to be set up by your program. To accomplish this, you should:

1. Code in your program the PWRSPL macro with TYPE=GEN or TYPE=UPD and specifying the operands

   ```
   JOB=jobname
   USER=userid
   ```

2. Use the available SPL DSECT to access the SPL.

For a list of the applicable SPL fields, see Figure 26 on page 74. For the lengths and data types of these fields, see the SPL DSECT that you get by a PWRSPL macro with TYPE=MAP; this DSECT gives additional explanations.

Legend: M = Mandatory; O = Optional

Name of Field	Applies to LST	PUN	Purpose/Contents
SPLORCFM	M	M	Record format
SPLGCL	O	O	Job (output) class
SPLGPW	O	O	Password
SPLDDP	O	O	Output disposition
SPLDPR	O	O	Output priority
SPLDSID	O	O	Output-system identifier
SPLDMOHP	O	O	Protect option: Hold (with disposition Y) when print/punch fails
SPLDUI	O	O	User information
SPLDTNN	O	O	Name of destination node
SPLDTUID	O	O	Name of destination user
SPLDPRGN	O	O	Programmer name
SPLDROOM	O	O	Room number
SPLDDEPT	O	O	Department number
SPLDBLDG	O	O	Building number
SPLDCREC	O	O	PUT-open restart record number
SPLDLREC	O	O	Maximum record length
SPLONCPY	O	O	Number of copies
SPLOCOMP	O		Name of compaction table
SPLOFORM	O	O	Form number
SPLOEWTR	O	O	External writer subsystem
SPLOFCB	O		Name of FCB-image phase
SPLOUCB	O		Name of UCB-image phase
SPLOUCBO	O		UCB options
SPLONSEP	O	O	Number of separator pages/cards
SPLEOPOF	O	O	Offset to OPTB area
SPLEOPLN	O	O	Length of passed OPTBs
SPLEOPTB	O	O	First (or only) OPTB

Figure 26 (Part 1 of 2). SPL Fields Applicable to a PUT-Output Service Request

Legend: M = Mandatory; 0 = Optional

Name of	Applies to		
Field	LST	PUN	Purpose/Contents
3200/3800 Specifications			
SPL3TAB1	0		Character-arrangement table 1
SPL3TAB2	0		Character-arrangement table 2
SPL3TAB3	0		Character-arrangement table 3
SPL3TAB4	0		Character-arrangement table 4
SPL3MODF	0		Copy-modification phase
SPL3CCHR	0		Character-arrangement table for copy-modification text
SPL3CPYG	0		Copy-group values
SPL3FLSH	0		Flash-ID
SPL3FLCT	0		Number of copies to be flashed
SPL3FLG1	0		Options byte (bit SPL3F138 must be set if any 3200/3800 option is specified)

Figure 26 (Part 2 of 2). SPL Fields Applicable to a PUT-Output Service Request

General Considerations

Format of Spool Records: Every output record that is to be spooled by VSE/POWER must have an eight-byte record prefix as shown in Figure 18 on page 43. A carriage-control character, if any, has to be inserted into this prefix. Special format data such as graphics is handled as is.

VSE/POWER does not check the validity of any carriage control character that might be associated with a spool record; it ignores the specification of carriage control characters for special-format data records.

The carriage control characters X'FF', X'FE', and X'FD' are reserved for use by VSE/POWER.

Page and Line Counts: For the records being spooled, VSE/POWER maintains page and line counts depending on the record type. The table in Figure 27 on page 76 shows how VSE/POWER maintains these counts.

Type of Records	Line Count	Page Count (see *)
With ASA	Incremented for each record.	Updated in accordance with carriage-control characters.
With MCC	Updated in accordance with carriage-control characters. X'00' and X'01' (write-no-space) is counted as a line.	Updated in accordance with carriage-control characters.
BMS, 3270 mapping, and CPDS	Incremented for each record.	Incremented for each page.
All others	Incremented for each record.	Set to 1
CPDS intermixed with records having ASA or MCC.	Incremented for a CPDS record. For non-CPDS records, see ASA- or MCC-type records, above.	Incremented for a CPDS record. For non-CPDS records, see ASA- or MCC-type records, above.
* Is set to 1 if, at the end of spooling, this count is still zero and the line count is 1 or greater.		

Figure 27. Line Counts as Maintained by VSE/POWER

VSE/POWER Account Records: VSE/POWER performs accounting for submitted output as follows:

- For output without a restart or a later expansion by an append operation, VSE/POWER's spool-record count is the same as for the output of a job submitted from a unit record input device.

- For output to be appended to an existing queue entry, VSE/POWER builds an extra set of spool-access account records:

  - Each time records are submitted in order to be appended, and
  - Only for the records submitted during the append operation.

- For output involving a restart, VSE/POWER counts the spooled output records only once. Assume, your program:

  1. Submits 1000 records for spooling.
  2. Requests a restart at record position 901.
  3. Submits another 200 records before it issues a Close request.

  VSE/POWER's record count then is 1100 records.

Verification SPLs: VSE/POWER returns a verification SPL:

1. After having successfully opened PUT-service processing.

   This SPL contains the same information that your program supplied in the request SPL, plus default values assigned by VSE/POWER for values not specifically supplied.

2. At the end of data submission for the queue entry.

   VSE/POWER passes this verification SPL in response to your Close request when submission of job output is complete or after having completed a segmentation request. In addition to the information supplied by your program, the SPL includes:

   - All of the VSE/POWER-generated job information such as job number and job suffix (segment number).
   - The default values used by VSE/POWER for values not specifically supplied by your program.
   - Statistics such as the total number of records spooled for your output.

   Checking this SPL can be of significance for spooling output. You need, for example, the VSE/POWER-assigned job number if data is to be appended to this queue entry or if spooling is to be restarted.

Handling an Abnormal-End Condition During PUT: If an abnormal-end occurs while VSE/POWER spools the output data, VSE/POWER's actions are as follows:

- For an abnormal end of your program or of the VSE/POWER service task:

  If the output is checkpointed, VSE/POWER retains the queue entry's spool data up to the last recorded checkpoint. The queue entry's disposition is X to avoid that another task can process the entry.

  If the output is not checkpointed, VSE/POWER deletes the currently processed queue entry, except as indicated below:

  - The failure occurred after successful completion of an open-restart request by VSE/POWER. In this case, the previously submitted data up to (but not including) the restart record still exists in the affected queue entry.

    VSE/POWER retains this queue entry with a disposition of X, and your program can set up the requested restart once more.

  - The failure occurred after successful completion of an open-append request by VSE/POWER. In this case, the data previously submitted (prior to the open-append request) still exists in the affected queue entry.

    VSE/POWER retains this queue entry with a disposition of X, and your program can set up a restart request.

  How to perform a restart is discussed under "Request a Restart" on page 86.

- For an abnormal end of VSE/POWER or of the VSE system:

  During VSE/POWER start-up, VSE/POWER searches the queue file for incomplete queue entries.

  - If the queue entry is checkpointed, VSE/POWER sets the spool pointer immediately behind the record last checkpointed. In addition, it adds the queue entry to the applicable class chain. When VSE/POWER start-up is complete, the queue entry is accessible for a restart request from your program or for printing if the central operator alters the entry's disposition.

  - If the queue entry is not checkpointed, VSE/POWER deletes this queue entry and the related space of the data file.

- For an abnormal end because of an I/O error on the data file:

  - If the output is checkpointed, VSE/POWER retains the queue entry's queue record and associated DBLK groups up to the last recorded checkpoint. The queue entry's disposition is set to X to avoid that another task can process the entry.

  - If the output is not checkpointed, VSE/POWER deletes the queue entry.

Coding Sequence

In general, the coding sequence for output submission is the same as for the submission of a job (or jobs) for queuing in an input queue. Your program issues an Open request, followed by a number of PUT-data requests, followed by a Close request and one or more message retrieval requests; your program can issue a quit request any time after a PUT-data request is complete. As mentioned earlier, this section deals primarily with output specific PUT requests.

Coding-sequence diagrams are given in context as each of these services is discussed.

Issue a Close-Service Request

Refer to Figure 28 on page 80, a coding-sequence diagram for a PUT-output Close request. You issue the request by setting byte PXUACT1 of your program's XPCCB to either of the following:

- The value equated to PXUATEOD – If no additional data is to be appended.

- The value equated to PXUATROE – If additional data is to be appended at a later point in time. Appending additional data is discussed under "Append Output to an Existing Spool File" on page 89.

Your program may pass the Close request in one of the following ways:

- Together with a null send buffer

  If you do this (after having successfully passed a send buffer containing data), your program must:

  1. Set byte PXUBTYP of the XPCCB to zero.

  2. Set up a null buffer. How to do this is shown in the section "Spool-Access Support Programming Example" on page 120 at the label GQUIT.

  3. Issue an XPCC FUNC=SENDR request after having set up the request.

- Together with a send buffer containing data records

  These records are the last output records spooled by VSE/POWER for the currently processed queue entry. The coding sequence at the label SDEOD in the section "Spool-Access Support Programming Example" on page 120 shows how to do this.

```
    Coding in your
    application program                        Comments
    -------------------              -----------------------------------
         |
         V
  WAIT  IJBXSECB                     Wait for the SENDR ECB to be posted
         |                           after the PUT-data request that is
         V                           to precede your Close request.
    Check the VSE/POWER return
     and feedback codes (in the
     XPCCB bytes PXPRETCD and
     PXPFBKCD, respectively).
         |
         V
  Close request
    XPCC  FUNC=SENDR,...             You can issue the request either:
         |                           - With the send buffer containing
         V                             data or an SPL.
    Check the return codes in        - With a null buffer being passed.
    register 15 and in the
    XPCCB (byte IJBXRETC).
         |
         V
  WAIT  IJBXSECB                     Wait for the SENDR ECB to be posted.
         |                           VSE/POWER passes a verification SPL
         V                           to your program's reply buffer.
    Check the reason code (in the
     XPCCB byte IJBXREAS).
         |
         V
    Gheck the VSE/POWER return and
     feedback codes (this is the
     same as above).
         |
         V
    Pick up and evaluate the veri-
     fication SPL, if necessary.
         |
         V
  End of service
```

Figure 28. Sequence Diagram - PUT-Output Close Request

- Together with a send buffer containing an update SPL

  If you do this (after having successfully passed a send buffer containing data), your program must:

  1. Set byte PXUBTYP of the XPCCB to PXUBTSPL.

2. Build the SPL in (or move it to) your program's send buffer.

   VSE/POWER analyzes the SPL and updates the control values for the currently processed output queue entry. This SPL is some kind of a last-minute change of the queue entry's job characteristics. However, VSE/POWER verifies only those of this SPL's fields which are listed in Figure 29; it ignores all other specifications passed by your program.

3. Issue an XPCC FUNC=SENDR request after having set up the request.

VSE/POWER returns a verification SPL to your program's reply buffer. This SPL includes the VSE/POWER assigned job number.

If the output's destination is another node, VSE/POWER spools this output into the XMT queue rather than into the local LST or PUN queue.

Field Name	Contents of Field
SPLGJB	The job name.
SPLGCL	The desired output class.
* SPLDDP	The output disposition.
* SPLDPR	The desired output priority.
SPLDSID	The identifier of the system that is to process the output (applies to a shared spooling environment; only the identifier of the VSE system is valid).
SPLDTNN	The name of the destination node.
SPLDTUID	The destination (remote) user ID.
SPLONCPY	The number of desired copies.
SPLOFORM	The form number to be used.

* Can be updated only if the submitted output is to be spooled into a local queue.

Figure 29. Update SPL Fields Verified by VSE/POWER

Request Output-Segmentation

Refer to Figure 30, a coding-sequence diagram for an output-segmentation request.

```
     Coding in your
     application program                       Comments
     ___________________         ____________________________________

          |
          V
  WAIT   IJBXSECB                Wait for the SENDR ECB to be posted
          |                      after the PUT-data request that is
          V                      to precede your output-segmentation
    Check the VSE/POWER return   request.
     and feedback codes (in the
     XPCCB bytes PXPRETCD and
     PXPFBKCD, respectively).
          |
          V
  Segmentation request
    XPCC  FUNC=SENDR,...         You can issue the request either:
          |                      - With the send buffer containing
          V                        data or an SPL.
    Check the return codes in    - With a null buffer being passed.
    register 15 and in the
    XPCCB (byte IJBXRETC).
          |
          V
  WAIT   IJBXSECB                Wait for the SENDR ECB to be posted.
          |                      VSE/POWER returns a verification SPL
          V                      to your program's reply buffer.
    Check the reason code (in the
     XPCCB byte IJBXREAS).
          V
    Check the VSE/POWER return and
     feedback codes (this is the
     same as above).
          V
    Pick up and evaluate the veri-
     fication SPL, if necessary.
          V
  PUT-Data Request               Continue after having filled your
    XPCC  FUNC=SENDR,...         program's send buffer again.
          |                      The first data record in your pro-
          |                      gram's send buffer goes into the
          V                      new output segment.
```

Figure 30. Sequence Diagram — Segmentation During PUT-Output Processing

Your program can request output-segmentation at any time after successful completion of a PUT-data request. You code this request in your program by setting byte PXUACT1 of the XPCCB to the value equated to PXUATSGM. You pass this request to VSE/POWER in one of the following ways:

- Together with a null send buffer

  If you do this, your program must:

  1. Set byte PXUBTYP of the XPCCB to zero.
  2. Set up a null buffer. How to do this is shown in the section "Spool-Access Support Programming Example" on page 120 at the label GQUIT.
  3. Issue an XPCC FUNC=SENDR request after having set up the request.

- Together with a send buffer containing data records

  This causes VSE/POWER to include the buffer's contents in the currently processed output segment. The contents of the next buffer that your program passes to VSE/POWER become part of the newly created output segment.

- Together with a send buffer containing an update SPL

  If you do this, your program must:

  1. Set byte PXUBTYP of the XPCCB to PXUBTSPL.
  2. Build the SPL in (or move it to) your program's send buffer.

     VSE/POWER analyzes the SPL and updates the control values for the currently processed output queue entry. This SPL is some kind of a last-minute change of the queue entry's job characteristics. However:

     - VSE/POWER verifies only those of the SPL's fields which are listed in Figure 29 on page 81; it ignores all other specifications passed by your program.
     - Any changed (or new) specifications that your program supplies in this update SPL are used by VSE/POWER also for the subsequent segment(s). If this is not desirable, your program has to pass another update SPL at the end of the next segment.

  3. Issue an XPCC FUNC=SENDR request after having set up the request.

Just like for a Close request, VSE/POWER returns a verification SPL after having successfully queued the segment. This SPL gives the

VSE/POWER assigned job-suffix (segment) number. VSE/POWER is then ready to accept further output for spooling into a new output segment.

The coding sequence in the section "Spool-Access Support Programming Example" on page 120 includes an output-segmentation request at the label PUTB2.

Request a Checkpoint

Consider requesting checkpoints to "save" the processing of records already passed to VSE/POWER should an abnormal-end condition occur. Your program can issue a checkpoint request before the first PUT-data request and after successful completion of any subsequent PUT-data request.

In processing a checkpoint request, VSE/POWER marks the queue entry as having been checkpointed and returns a checkpoint-response record. This response record contains the VSE/POWER-recorded number of the record spooled for the queue entry just before the checkpoint was taken. For the layout of a checkpoint-response record, see the DSECT at the label PXCRDSECT.

The number of the checkpointed record may not be the same as the number of this record according to your program's own record count. Therefore, your program should:

1. Relate the checkpoint record number to the corresponding record number of the program's own count.
2. Save this relation for a later restart, should this become necessary.

By relating this number to your own program's record count, you can synchronize your program's output with the record count maintained by VSE/POWER.

Refer to Figure 31 on page 85, a coding-sequence diagram for a checkpoint request. In your program, you code this request as follows:

1. Set byte PXUACT1 of the XPCCB to the value equated to PXUATCHK.
2. Make a reply buffer available.
3. Pass the request to VSE/POWER. To do this, issue an XPCC FUNC=SENDR request with either of the following:
   - Data contained in your program's send buffer. In this case, VSE/POWER spools that buffer's contents first and then processes the checkpoint request.
   - A null send buffer. This requires that your program:
     a. Sets byte PXUBTYP of the XPCCB to zero.
     b. Sets up a null buffer (how to do this is shown in the section "Spool-Access Support Programming Example" at the end of this chapter – at label GQUIT, for example).

```
   Coding in your
   application program                    Comments
   ___________________                 _______________________________

        |
        V
WAIT   IJBXSECB                        Wait for the SENDR ECB to be posted
        |                              after the PUT-data request that is
        V                              to precede your checkpoint request
  Check the VSE/POWER return
  and feedback codes (in the
  XPCCB bytes PXPRETCD and
  PXPFBKCD, respectively).
        |
        V
Checkpoint request
  XPCC   FUNC=SENDR,...                You can issue the request either:
        |                              - With the send buffer containing
        V                                data or an SPL.
  Check the return codes in            - With a null buffer being passed.
  register 15 and in the
  XPCCB (byte IJBXRETC).
        |
        V
WAIT   IJBXSECB                        Wait for the SENDR ECB to be posted.
        |                              VSE/POWER passes a checkpoint re-
        V                              sponse record to your program's re-
  Check the reason code (in the        ply buffer.
  XPCCB byte IJBXREAS).
        |
        V
  Check the VSE/POWER return and
  feedback codes (this is the
  same as above).
        |
        V
PUT-data request                       Continue after having filled your
  XPCC   FUNC=SENDR,...                program's send buffer again.
        |
        V
  End of service
```

Figure 31. Sequence Diagram – Checkpoint for PUT-Output Processing

Request a Restart

VSE/POWER permits your program to request a restart as follows:

- During PUT-output processing, behind a previously spooled record.

  This restart causes the specified restart record and all subsequent records spooled previously to be overwritten.

  A restart during processing can be risky. Your program's record count (if maintained) may be different from that of VSE/POWER because VSE/POWER inserts an additional record whenever a write-and-skip to channel 1 occurs. Section "Request a Checkpoint" on page 84 indicates how your program can use VSE/POWER's checkpoint-response records to keep track of suitable restart points.

- Together with or immediately after an Open request for an existing queue entry.

  As the restart point, you can specify 0 (or nothing). In this case, VSE/POWER sets its restart pointer immediately behind the last record in the queue entry's data file. For a checkpointed queue entry with disposition X, this is the record last checkpointed by VSE/POWER.

  Specifying 0 may be risky. If a system or program failure occurs after VSE/POWER has passed a recorded checkpoint and before your program could record this checkpoint, then VSE/POWER and your program are not synchronized.

  To avoid problems, you can specify a suitable restart point as recorded by your program. VSE/POWER indicates in its verification SPL the corresponding, check-pointed record count.

  In case of a restart, VSE/POWER examines a specified restart point. If this point:

  - Is higher than the last recorded checkpoint, VSE/POWER accepts this restart point as specified.

  - Is equal to or lower than the last recorded checkpoint, VSE/POWER lowers the checkpoint value to the restart value, minus 1, and notifies your program of the change (return/feedback code = PXPRCOK/PXPOOCIA).

Restart During PUT-Output Processing: If, in its restart control record, your program specifies a restart record number lower than or equal to the logical record last checkpointed, then VSE/POWER:

1. Positions the spool pointer as requested, just as if the queue entry were not checkpointed.

2. Records the specified restart record number (minus one) as the new checkpoint-record number.

   VSE/POWER returns to your program a checkpoint-response record together with applicable return and feedback codes. The response record confirms to your program the newly recorded checkpoint. For the layout of a checkpoint-response record, see the DSECT generated by PWRSPL TYPE=MAP at the label PXCRDSCT.

Refer to Figure 32 on page 88, a coding-sequence diagram for a restart request.

```
    Coding in your
    application program                     Comments
    ___________________                     ________________________________

          |
          V
  WAIT   IJBXSECB                           Wait for the SENDR ECB to be posted
          |                                 after the PUT-data request that pre-
          V                                 cedes your restart request.
    Check the VSE/POWER return
    and feedback codes (in the
    XPCCB bytes PXPRETCD and
    PXPFBKCD, respectively).
          |
          V
  Restart request
    XPCC  FUNC=SENDR,...                    Your program's send buffer must
          |                                 contain a restart control record.
          V
    Check the return codes in
    register 15 and in the
    XPCCB (byte IJBXRETC).
          |
          V
  WAIT   IJBXSECB                           Wait for the SENDR ECB to be posted.
          |                                 VSE/POWER may pass a checkpoint-
          V                                 response record to your program's
    Check the VSE/POWER return and          reply buffer.
    feedback codes (this is the
    same as above).
          |
          V
    Pick up and evaluate the check-
    point response record, if this
    is applicable.
          |
          V
  PUT-data request                          At this point, the coding sequence
    XPCC  FUNC=SENDR,...                    is the same as for the submission of
          |                                 data records for spooling.
          V
    End of service
```

Figure 32. Sequence Diagram — Restart for PUT-Output Processing

To set up and pass the request to VSE/POWER, your program must:

1. Set byte PXUBTYP of the XPCCB to the value equated to PXUBTCTL. This indicates that your program's send buffer contains a control record.

2. Set up a restart control record in your program's send buffer. For the layout of this record, see the DSECT at label PXRSDSCT. The record specifies the number of the logical record at which output spooling is to be resumed.

3. Issue an XPCC macro with FUNC=SENDR.

Restart with an Open Request: This kind of a restart applies if output spooling is to be restarted because, for example, an abnormal-end condition had occurred.

A restart with an Open request is possible if the following is true:

- The applicable queue entry is queued with one of the dispositions D, H, K, L, and X.

- The requestor is the owner (originator) of the queue entry.

If your program does not pass a restart-record number, then:

- For a queue entry with disposition X, VSE/POWER positions the spool pointer behind the entry's last checkpointed record.

- For a queue entry with a disposition other than X, VSE/POWER positions this pointer to the end of the entry's data file.

If your program passes a restart-record number, it should provide for a routine verifying that VSE/POWER's record count and your program's record count are synchronized. How you can do this is indicated under "Restart During PUT-Output Processing" on page 86.

Your program requests the desired restart by issuing an XPCC macro with FUNC=SENDR and passing to VSE/POWER a restart SPL (MODE=RESTART specified in the PWRSPL macro). In the SPL, certain fields are to be updated as listed in Figure 33 on page 90. VSE/POWER confirms the request in the same way as it confirms a normal open PUT-service request: by passing a verification SPL to your program.

Append Output to an Existing Spool File

VSE/POWER permits additional data to be appended to (added at the end of) an existing output queue entry if your program:

1. Is the owner (originator) of this queue entry.

2. Closed the original spool request with the append-option bit PXUATROE set in byte PXUACT1 of its XPCCB.

Legend: M = Mandatory; O = Optional

Name of Field	Applies to LST	PUN	Purpose/Contents
SPLGFB1	M	M	Set restart function
SPLGCL	M	M	Job (output) class
SPLGJB	M	M	Job name
SPLGJN	M	M	Job number
SPLGUS	M	M	User identifier
SPLGQI	M	M	Queue identifier
SPLGRQB	M	M	Request type (PUT)
SPLGJS	O	O	Job suffix
SPLGOPT	O	O	Set no-wait option
SPLGPW	O	O	Password
SPLDCREC	O	O	PUT-open restart record number

Figure 33. SPL Fields to be Updated – Open-Restart Request for Output

3. Issues an open-append request (which reinitiates PUT-service processing for the queue entry) by passing to VSE/POWER an SPL that specifies the append option. This SPL should contain the values passed by VSE/POWER in its original verification SPL in the fields listed in Figure 34 on page 91. You may find it convenient to have your program save the verification SPL and use it as request SPL for the append request.

Legend: M = Mandatory; O = Optional

Name of Field	Applies to		Purpose/Contents
	LST	PUN	
SPLGFB1	M	M	Set append function
SPLGCL	M	M	Job (output) class
SPLGJB	M	M	Job name
SPLGJN	M	M	Job number
SPLGUS	M	M	User identifier
SPLGQI	M	M	Queue identifier
SPLGRQB	M	M	Request type (PUT)
SPLGJS	O	O	Job suffix (segment number)
SPLGOPT	O	O	Set no-wait option
SPLGPW	O	O	Password

Figure 34. SPL Fields to be Updated – Open-Append Request for Output

VSE/POWER confirms the request in the same way as it confirms a normal open PUT-service request: by passing a verification SPL to your program. After having passed this SPL, VSE/POWER is ready to accept PUT-data requests from your program.

Specify Output Parameter Text Blocks (OPTBs)

In your program you can specify one or more Output Parameter Text Blocks (OPTBs) when describing the characteristics of the output queue entry being passed to VSE/POWER. An OPTB represents a user-defined keyword (in a * $$ LST or * $$ PUN statement) that has been defined in an autostart DEFINE statement.

An OPTB is structured as a sequence of text units. The number and sequence of text blocks is arbitrary. The format of the OPTBs is shown in Figure 35 on page 92.

```
        ┌────┬────┬────┬────────────────┬────┬────────────────┬──//──┐
        │ ID │ CC │ LL │  Data element  │ LL │  Data element  │      │
        └────┴────┴────┴────────────────┴────┴────────────────┴──//──┘
Bytes:    2    2    2          n          2          n
```

ID Registered (unique) keyword identifier

CC Count of the data elements supplied for the keyword parameter. The valid range is from 0 to 16,383. A count of 0 indicates either a missing positional or defaulted parameter. In this case, no data elements should follow the count field.

LL Length of the data element (keyword parameter value). The valid range is from 0 to 16,383. A length of 0 indicates a null value.

Figure 35. Output Parameter Text Block (OPTB) Format

Passing OPTBs to VSE/POWER

The OPTBs are passed to VSE/POWER as an appendage of the SPL at PUT Open time. As shown in Figure 36, the SPL contains two 2-byte fields indicating the total length of the OPTB area and the offset to this area. A length of zero indicates that no such area exists.

```
                                   |<────── OPTB Area ──────────>|
                                   |                             |
┌──────────────┬─────────┬─────────┼───────┬───────┬──//──┬──────┤
│  Normal SPL  │SPLEOPOF │SPLEOPLN │ OPTB1 │ OPTB2 │      │ OPTBn│
├──────────────┴─────────┴─────────┼───────┴───────┴──//──┴──────┤
|                                  |                             |
|<───Length of normal SPL────────->|<──── Length of all OPTBs ──>|
|                                                                |
|<──────────────────── Length of total SPL ─────────────────────>|
```

where: SPLEOPOF = Two-byte field containing the offset from the beginning of the SPL to the OPTB area
SPLEOPLN = Two-byte field containing the length of the OPTB area

Figure 36. Spool Parameter List Structure

If one or more OPTBs are appended to the SPL, VSE/POWER checks the OPTBs for correct specification. An OPTB representing a keyword which is not defined within VSE/POWER is taken as is (see also the autostart DEFINE statement in VSE/POWER Installation and Operations Guide). If all OPTBs are valid, VSE/POWER builds an output processing section and includes it in the data set header record (DSHR).

The total length of all OPTBs, including the length of all other sections (such as the general or the 3800 section) present in the DSHR may not exceed 32.760 bytes.

Note: All OPTBs which are of the type binary must have the same length as specified in the appropriate DEFINE statement.

Check the Return Information

For the return information to be checked by your program after an XPCC request, refer to the section "XPCC Macro" on page 113.

For each individual PUT-output type service request, your program should also check the return information supplied by VSE/POWER. Provide for this checking after your program's SENDR ECB has been posted.

Figure 37 lists the return and feedback codes that VSE/POWER may supply when it processes a PUT-service related request for the submission of output data. The list is ordered in ascending order by code values; it relates the codes to the applicable request types; it gives the names that are equated to the feedback codes.

A complete list of the VSE/POWER return and feedback codes is given in the DSECT PXPUSER, which the assembler generates for a PWRSPL TYPE=MAP macro. You find the return codes at label PXPRETCD and the feedback codes at label PXPFBKCD.

			Request Type									
Mnemonic	Ret. Code	Fdbk Code	PUT Open	PUT Data	Check-point	Re-start	Seg-ment	Close	Quit	Get Msg	Get OPTB	Mod. OPTB
PXP00OK	00	00	X	X	X	X	X	X	X	X	X	X
PXP00EOD		01								X		
PXP00NRS		03			X		X	X	X			
PXP00RTR		04		X				X				
PXP00ZBF		05		X								
PXP00CIA		06				X						
PXP04NOF	04	01	X									
PXP04JOP		02	X									
PXP04IDP		05	X									
PXP04RER		06				X						
PXP04SOD		08	X	X			X	X				
PXP04SOA		09	X	X			X	X				
PXP04BER		0A										X
PXP04NMU		0D	X									
PXP04WDP		0E	X									
PXP04JSR		0F	X									
PXP04ONF		11									X	X

Figure 37 (Part 1 of 3). Return and Feedback Codes for PUT-Output Service Requests

			Request Type									
Mnemonic	Ret. Code	Fdbk Code	PUT Open	PUT Data	Check-point	Re-start	Seg-ment	Close	Quit	Get Msg	Get OPTB	Mod. OPTB
PXP08SPL	08	01	X									
PXP08REQ		02	X									
PXP08JNM		05	X				X	X				
PXP08QID		06	X									
PXP08CLS		07	X				X	X				
PXP08PWD		08	X									
PXP08UID		09	X									
PXP08RFM		0A	X									
PXP08DSP		0B	X				X	X				
PXP08PRY		0C	X				X	X				
PXP08SID		0D	X				X	X				
PXP08TNN		0E	X				X	X				
PXP08TUN		0F	X				X	X				
PXP08FNO		10	X				X	X				
PXP08FCB		11	X									
PXP08UCB		12	X									
PXP08FLH		14	X									
PXP08CPT		15	X									
PXP08CGP		16	X									
PXP08CHR		17	X									
PXP08MOD		18	X									
PXP08CCR		19	X									
PXP08BTS		1A								X	X	
PXP08IAB		1C			X		X	X	X	X		
PXP08ICR		1D			X	X					X	X
PXP08PRG		1E	X									
PXP08ROO		1F	X									
PXP08DPT		20	X									
PXP08BLD		21	X									
PXP08CON		22	X	X	X	X	X	X	X	X		
PXP08ROL		23		X								
PXP08IBT		24	X	X	X	X	X	X	X	X		
PXP08ROS		25						X	X			
PXP08SOS		26	X	X	X	X	X	X		X		
PXP08BOS		27	X					X	X	X		

Figure 37 (Part 2 of 3). Return and Feedback Codes for PUT-Output Service Requests

			Request Type									
Mnemonic	Ret. Code	Fdbk Code	PUT Open	PUT Data	Check-point	Re-start	Seg-ment	Close	Quit	Get Msg	Get OPTB	Mod. OPTB
PXP08RPW	08	2A		X				X				
PXP08FB1		2B	X									
PXP08IML		2C	X									
PXP08IEX		2D	X									
PXP08SPA		2E		X				X				
PXP08ICC		2F		X				X				
PXP08IRR		38									X	X
PXP08IOP		39	X									X
PXP08OLM		3A										X
PXP08DOP		3B	X									
PXP08OTL		3C	X									
PXP08IDH		3D									X	X
PXP0CINS	0C	01	X	X	X	X	X	X	X	X		
PXP0CIXF		02	X	X	X	X	X	X	X	X		
PXP0CBTL		03	X									
PXP0CIOE		07	X	X	X	X	X	X	X	X		
PXP10PSP	10	05	X	X	X	X	X	X	X	X		
PXP10SIE		06	X	X	X	X	X	X	X	X		

Figure 37 (Part 3 of 3). Return and Feedback Codes for PUT-Output Service Requests

END ACCESS TO VSE/POWER

To end accessing VSE/POWER services via a communication path, this path is to be removed. This can be done either by a request from your program or by a request from VSE/POWER. A request from your program is indicated when there is no need for further access requests via a certain communication path and VSE/POWER has finished processing the last access request.

End of Access Requested by Your Program

Coding Sequence: Refer to Figure 38 on page 96, a coding-sequence diagram for the removal of a communication path from within your program. The section "Spool-Access Support Programming Example" on page 120 includes a disconnect and terminate coding sequence at labels DISCT and TERMN, respectively.

Check the Return Information: Your program should check the return codes set by the system on completion of the XPCC macro request. This ensures an orderly termination processing. These macro-return codes are listed and briefly described under "XPCC Macro" on page 113.

```
    Coding in your
    application program                 Comments
    -------------------                 --------------------------------

         |
         V
 WAIT  IJBXSECB                         Wait for the SENDR ECB to be posted
         |                              after your program's last access-
         V                              service request.
   Check the VSE/POWER return
    and feedback codes (in the
    XPCCB bytes PXPRETCD and
    PXPFBKCD, respectively).
         |
         V
 Disconnect request
   XPCC FUNC=DISCONN,...                Following this request, the communi-
         |                              cation path set up in your program
         V                              by XPCC FUNC=CONNECT is no
   Check the return codes in            longer available. To set up the
   register 15 and in the               path again, should this be desir-
   XPCCB (byte IJBXRETC).               able, issue an XPCC request with
         |                              FUNC=CONNECT.
         V
 Terminate request                      This is some sort of a log off by
   XPCC FUNC=TERMIN,...                 your program. To set up the path
         |                              again, should this be desirable,
         |                              start out with an XPCC request
         |                              specifying FUNC=IDENT.
         V
   Check the return codes in            This ensures orderly discontinuation
   register 15 and in the               of using the spool-access support.
   XPCCB (byte IJBXRETC).
```

Figure 38. Sequence Diagram - End Access to VSE/POWER

End of Access Requested by VSE/POWER

VSE/POWER indicates this condition by return and feedback codes in field IJBXRUSR of the XPCCB. A complete list of the VSE/POWER return and feedback codes is given in the DSECT PXPUSER, which the assembler generates for a PWRSPL TYPE=MAP macro. You find the return codes at label PXPRETCD and the feedback codes at label PXPFBKCD.

SET UP TWO OR MORE COMMUNICATION PATHS

You may, if this is desirable, have your program set up two or more communication paths to VSE/POWER. To do this, proceed as follows after having identified your program (by an XPCC macro with FUNC=IDENT as described in section "Set Up a Communication Path" on page 33):

1. Copy the XPCCB used for identification

   On successful completion of the request, the VSE system returns an X (= cross partition) identifier in field IJBXIDK of your XPCCB. The system expects an XPCCB with this identifier for a subsequent XPCC request with FUNC=CONNECT. It follows then that your program needs a copy of the XPCCB with the returned X identifier for every communication path which is to be set up.

2. Issue an XPCC request with FUNC=CONNECT

   The system provides a uniquely identified communication path by inserting a P (= path) identifier in field IJBXPID of the XPCCB you use.

   For any additional communication path that you want to set up, issue a new XPCC request with FUNC=CONNECT. Each of these requests must use a new copy of the XPCCB which you used for identification.

   A connect request must be complete before you can issue the next one.

DESCRIPTION OF THE SPOOL-ACCESS SUPPORT MACROS

For each of the described macros, the information is given in applicable sections as follows:

1. A short summary of the macro's purpose.
2. The macro's format as used for access to VSE/POWER services.
3. A description of the macro's operands.
4. Possible return codes (applies only to the XPCC macro).

For the meaning of symbols used in showing the formats of the macros, refer to Chapter 1 of VSE/POWER Installation and Operations Guide. Continuation codes that may be required in column 72 are not shown as part of the macro formats.

The codes given under the heading "Return Information" represent hexadecimal values.

MAPXPCCB Macro

The macro causes a DSECT of the XPCCB to be generated. It has no operands.

Format of the Macro

Name	Operation	Operands
[name]	MAPXPCCB	

For name, you may assign to the DSECT a label of your own choosing.

The macro has no operands.

PWRSPL Macro

You can use the macro to do one of the following:

- Generate a spool parameter list (SPL).
- Update an SPL.
- Generate DSECTs of the SPL and of the various request control records and VSE/POWER-response records.

For complex applications, the macro may not offer the scope of required control. The macro does not offer the required scope of control for applications involving the submission of output.

If the macro's scope does not meet your application's requirements, provide for setting up certain fields of the SPL by coding of your own. You do this by accessing the applicable SPL (field) via a generated DSECT.

Format of the Macro

Name	Operation	Operands
[name]	PWRSPL	TYPE={GEN|MAP|UPD} [,CLASS={class|(reg)}] [,FUNC={(ALTER,attrib-type)| CANCEL| COMMAND| DELETE| DISPLAY| HOLD| RELEASE}] [,JOBN={jobname|(reg)}] [,JNUM={fieldname|(reg)}] [,JSUF={fieldname|(reg)}] [,MODE={APPEND| BROWSE| GENERIC| RESET| RESTART}] [,NEWVAL={field|(reg)}] [,OPT={RESET| ([ALLCPY][,CTLREC][,FORMAT] [,NOWAIT][,RETSEP])}] [,PRFX=xxx] [,PWD={password|(reg)}] [,QUEUE={RDR|LST|PUN|XMT}] [,REQ={CTL|GET|PUT}] [,SPL={splname|(reg)}] [,USERID={userid|(reg)}]

Description of Operands

TYPE={GEN|MAP|UPD}

The operand specifies the desired type of macro expansion:

GEN Causes an SPL to be generated.

MAP Causes a DSECT of the SPL to be generated. Only the operand PRFX=xxx is meaningful together with a TYPE=MAP specification.

UPD Requests that an SPL defined in the program by PWRSPL TYPE=GEN be updated in accordance with the specified operands. SPL fields corresponding to omitted operands remain unchanged.

CLASS={class|(reg)}

The operand specifies the class that:

- Matches the class of the queue entry which is to be retrieved (REQ=GET is specified).
- Is to be assigned to the output queue entry (REQ=PUT is specified).
- Is to be used as a search argument if a control function is to be processed (REQ=CTL is specified).

As class value, specify the desired input or output class – one of those defined for your location. The class values 0 through 9 (partition related) are valid only for manipulating the RDR or the XMT queue.

If the macro specifies TYPE=GEN, then:

- You cannot use register notation.
- Class A is used as default.

If you use register notation (together with TYPE=UPD), the specified register must point to a one-byte field that contains the class value.

FUNC={(ALTER,attrib-type)|CANCEL|COMMAND|DELETE|DISPLAY|HOLD|RELEASE}

The operand applies only if you specify also REQ=CTL; it specifies the type of function to be performed.

ALTER

Causes VSE/POWER to alter, for a job (or group of jobs) in the accessed queue, the attribute that you specify for attrib-type.

For attrib-type, you can specify one of the attributes discussed under NEWVAL={field|(reg)}, below. Only one attribute can be changed per CTL request.

CANCEL

Causes VSE/POWER to cancel (flush) the job identified by job name and, optionally, by job number.

COMMAND

Indicates that VSE/POWER is to process the command supplied in the field SPLCFLD of the SPL that is being generated or updated. VSE/POWER accepts the command without error checking for the command.

For passing a command to VSE/POWER, the following rules have to be observed:

- The command must be set up using upper-case letters.
- The command cannot be longer than 72 bytes.
- Continuation of the command is not supported.
- At least one blank must follow the command verb (such as PDISPLAY or just D).

The table below lists the commands that VSE/POWER accepts via a spool-access communication path:

```
PALTER jobname (see Note 1 below)
PBRDCST
PCANCEL jobname (see Note 1 below)
PDELETE jobname (see Note 1 below)
PDELETE MSG
PDISPLAY jobname (see Note 1 below)
PDISPLAY MSG
PDISPLAY A
PDISPLAY T
PDISPLAY PNET
PDISPLAY TAPE (see Note 2 on page 102 below)
PDISPLAY VIO (see Note 2 on page 102 below)
PFLUSH DEV (see Note 2 on page 102 below)
PGO DEV (see Note 2 on page 102 below)
PHOLD jobname (see Note 1 below)
PINQUIRE
PRELEASE jobname (see Note 1 below)
PRESTART DEV (see Note 2 on page 102 below)
PSETUP DEV (see Note 2 on page 102 below)
PSTART DEV (see Note 2 on page 102 below)
PSTOP DEV (see Note 2 on page 102 below)
PXMIT nodeid
PXMIT DEV (see Note 2 on page 102 below)
```

Notes:

1. Accepted only if there is a match of the recorded and specified user-IDs (origin or target) and, if applicable, also these passwords.

   Only the owner of an entry can manipulate a target entry with a destination of ANY (unless his own userid is ANY which, however, is not recommended).

2. Only for authorized users – For example the subsystem administrator, if there is one.

DELETE
Causes VSE/POWER to delete the named job from the specified queue.

DISPLAY
Causes VSE/POWER to return information about the queue entries as described by one or a combination of the following: job name, job number, and class.

HOLD
Causes VSE/POWER to change the disposition of one or more jobs to:

H (hold) if it was D (dispatchable)
L (leave) if it was K (keep).

RELEASE
Causes VSE/POWER to take one or more jobs out of the hold or leave state and make them available for processing.

JOBN={jobname|(reg)}
The operand specifies the VSE/POWER job name that is to be used for the execution of the request. The job name you specify must be alphameric and not longer than eight characters.

If the macro specifies TYPE=GEN, then:

- You must define the name in your program as a character constant.
- Omission of the operand causes AUTONAME to be used as the default name.

If you code this operand together with TYPE=UPD, specify for jobname the label of an eight-byte field that contains the job name left justified and padded with blanks.

If you use register notation, the specified register must point to an eight-byte field that contains the name.

JNUM={fieldname|(reg)}
The operand can be used only together with TYPE=UPD. It specifies the number which VSE/POWER assigned to the job whose queue entry is to be manipulated or whose data is to be retrieved.

If you use register notation, the specified register must contain the job number. If you do not use register notation, then fieldname must be the label of a half-word that contains the job number in binary notation.

If you do not want to supply a job-number, set the field (or register) to binary zeros.

JSUF={fieldname|(reg)}

The operand can be used only together with TYPE=UPD. It specifies the job-suffix (segment) number assigned to the queue entry that is to be manipulated or to be retrieved.

If register notation is used, the specified register must contain the number. If you do not use register notation, then fieldname must be the label of a half-word containing the number (in binary).

MODE={APPEND|BROWSE|GENERIC|RESET|RESTART}

The operand specifies the mode of operation for the requested service:

APPEND

Spooling is to continue at the end of an already existing queue entry. This applies only to the PUT output function.

BROWSE

Useful primarily if you intend to examine (but not to update) a queue entry.

If you specify BROWSE, you must also provide the name of the job to be accessed, with or without the applicable job number and job suffix. A queue entry accessed in BROWSE mode is left unchanged in its queue even if the entry's disposition is D.

For a retrieval in BROWSE mode, VSE/POWER accepts only a subset of GET-service requests (with an action code in byte PXUACT1 of the XPCCB field IJBXSUSR) as shown below:

Type of Request	Mnemonic Equated to the Action Code
A retrieval request	PXUATSDR
A quit request	PXUATABR
A restart request	Not applicable (see *)

* You submit this request by passing (to VSE/POWER) a restart-control record.

GENERIC

Causes VSE/POWER to retrieve the first eligible queue entry destined for a certain user within the specified class. When processing a retrieval request in this mode, VSE/POWER ignores the specification of a job name, a job number, or a job suffix.

VSE/POWER selects, for retrieval, the queue entry whose characteristics are closest to the ones defined in the PWRSPL macro.

You can include in your request up to three additional classes by:

1. Inserting the additional classes left justified in the field SPLGNV of your SPL followed by a blank.
2. Setting the flag SPLGOACL in byte SPLGOPT, the SPL's option byte.

RESET
Causes the mode settings to be reset to the default values.

RESTART
Spooling is to begin at a certain record of an already existing queue entry (PUT-output function). This record is either of the following:

- The one whose number your program supplies in field SPLDCREC of the applicable SPL.
- The record last checkpointed by VSE/POWER if your program does not supply a record number in this SPL field.

NEWVAL={field|(reg)}
The operand names the field that contains the new value to be used by VSE/POWER as attribute for the named queue entry. This value must be defined as a character constant if you use the operand together with TYPE=GEN.

The meaning of the value depends on your specification in the FUNC=(ALTER,attrib-type) operand. This meaning is given below by alter attribute:

CLASS
The name of a one-byte field that contains the new class of the queue entry. If you use register notation, the register must point to the one-byte field.

CMPACT
The label of a four-byte field which contains the name of the new compaction table set to be used for transmitting the queue entry to an SNA work station. Supply this table set's name left justified without leading blanks.

Instead of the name of a compaction table, you may specify either:

* To indicate that the default compaction table is to be used.
NO To indicate that no compaction should be performed.

If you use register notation, the register must point to the four-byte field.

COPY
The name of a three-byte field that contains, in character format, the new number of copies (any value from 1 to 255). If you supply the number right justified, leading zeros are required.

If register notation is used, the register must point to the three-byte field.

The specification applies only to output queue entries; it is ignored if you specify it for an input queue entry.

DISP
The name of a one-byte field that contains, in character format, the new disposition (D, K, H, or L) of the affected queue entry. If you use register notation, the register must point to the one-byte field.

For further information on disposition refer to the VSE/POWER Installation and Operations Guide.

How the operator is to handle dispositions X and Y is described in the Chapter "Operating with VSE/POWER" of that manual.

NODE
The label of an eight-byte field that contains, in character format, the new target destination of the queue entry. In this field, supply the destination left justified without leading blanks. If you use register notation, the register must point to the eight-byte field.

PRI The name of a one-byte field that contains, in character format, the new priority. If you use register notation, the register must point to the one-byte field.

REMOTE
The label of a three-byte field that contains, in character format, the new remote identifier. This is a value from 0 to 250.

If you supply the number right justified, leading zeros are required. If you use register notation, the register must point to the three-byte field.

The specification applies only to output queue entries; it is ignored if you specify it for an input queue entry.

SYSID
The label of a one-byte field that contains, in character format, the new system identifier. If you use register notation, the register must point to the one-byte field.

USER
The label of an eight-byte field that contains, in character format, the new target user-ID of the queue entry. In that field, supply this ID left justified without any leading blanks. If you use register notation, the register must point to the eight-byte field.

OPT={RESET|([ALLCPY][,CTLREC][,FORMAT][,NOWAIT][,RETSEP])}
The operand specifies options for performing the requested service.

You may omit the enclosing parentheses if you specify only one of the options. Do not code a comma preceding your first option of your list.

If a specified option does not apply to the requested function, VSE/POWER ignores this option.

RESET
Causes VSE/POWER to reset (turn off) any option specified previously.

ALLCPY
Causes VSE/POWER to return all copies of an output queue entry to the requestor. The specification applies only if you specified REQ=GET.

Each copy of the queue entry starts with record number 1 and an SPL that describes the copy.

CTLREC
Causes VSE/POWER to return also immediate control records (such as skip to channel 1 and space 2 lines) when retrieving an output queue entry. A control record consists of a record prefix and one byte of data. The command code is contained in the record prefix.

FORMAT
Causes the result of a requested queue-related display to be returned as fixed format records rather than free-format messages.

The specification applies only if you specify REQ=CTL together with the QUEUE operand; it is effective only on the local queues.

If you specify PWRSPL with TYPE=MAP, the assembler generates a DSECT of the fixed-format display message (label PXFMDSCT) into your program.

If you request a queue display in free format, VSE/POWER returns the requested information in the form of console messages.

NOWAIT

Requests control to be returned to the requestor when a wait condition occurs because of lack of disk space. If you do not specify NOWAIT, VSE/POWER waits for such space to become available. What this means for your program is discussed below.

During GET-service processing, VSE/POWER may find that the account-file is full. The situation may come up when VSE/POWER executes one of the following subfunctions:

- Close the processing for the currently accessed queue entry.
- Perform a FLUSH-HOLD for the spool request (applies only to an application involving external device support).
- Purge the involved queue entry.
- Quit processing the request.

When the account-file-full condition occurs, the function has already been performed. Therefore, VSE/POWER cannot inform your program right away. If no other GET (CTL or PUT) request follows, your program does not become aware of this situation. However, a subsequent GET (CTL or PUT) request from your program is rejected with applicable return and feedback codes supplied by VSE/POWER. In your program, you can then decide, whether you want to wait and retry after a certain time interval or to setup a new communication path to VSE/POWER in order to start the new function.

During a PUT-service processing, VSE/POWER may run into a "short of disk space" situation as indicated:

- While VSE/POWER is spooling a job or job output.

  VSE/POWER stops further spooling of the submitted job or output. This results in the following:

  - If an output file is being spooled without being checkpointed, this file is lost, and VSE/POWER queues an information message. If the file is checkpointed, VSE/POWER queues the file up to last committed checkpoint; the file's remaining data is lost.

- If a single job is being spooled, the job is lost, and VSE/POWER queues an information message.

- If multiple jobs are being spooled, the jobs already spooled are kept in the input queue, but the job being processed and all subsequent ones are lost. VSE/POWER queues a message and returns a verification SPL for the last job spooled successfully.

  VSE/POWER ignores input that is being read from a diskette as a result of the processing of a * $$ RDR statement in the failing job. However, the diskettes are fed as for normal processing until the end of this diskette (SYSIN) input is reached.

- If a segment-output request is being processed, VSE/POWER may or may not pass to your program a verification SPL in addition to the data-file-full indication.

  By passing this SPL, VSE/POWER informs your program that all data submitted up to this point has been spooled and a queue entry for the segment exists. Processing for building another segment cannot continue.

  If VSE/POWER passes only the data-file-full indication, all data submitted since the last successful segment request or checkpoint (whichever applies) is lost.

- When VSE/POWER tries to write into the account file.

  This can occur after a Close, quit, or segment request; it can occur after a spool-data request during multiple-job submission.

  - For a Close or quit request, VSE/POWER returns a successful completion indication. However, VSE/POWER rejects any subsequent PUT, GET or CTL function as long as the account-file-full situation exists. For the new function request, VSE/POWER performs no error checking; instead it returns to your program the return- and feedback-code combination PXPRCOKF and PXPS04SOA to indicate that the account file is full.

- For a segment request, VSE/POWER returns the PXPRCOKF/PXPS0S0A return/feedback-code combination together with the SPL that describes the output segment just queued. VSE/POWER does not accept any further spool requests.

- For multiple-job submission (with one open PUT-service request), VSE/POWER returns a verification SPL together with the PXPRCOKF/PXPS04S0A return/feedback code combination. This SPL applies to the job that was queued, but for which no account record could be written. Any subsequent job-spool requests are rejected by VSE/POWER.

  Your program may submit an * $$ RDR statement and effect multiple-job submission via diskette input. An account-file full condition during the processing of an end-of-job statement causes VSE/POWER to ignore all subsequent diskette data and the data behind the * $$ RDR statement in your program's send buffer.

RETSEP
Causes separator pages (or cards) to be returned as normal data records in front and at the end of the requested output queue entry. Separator pages (cards) that VSE/POWER builds are passed with their MCCs.

The option applies only if you specified REQ=GET and if a JSEP value other than zero was specified for the queue entry.

PRFX=xxx
Use this operand if, for the generated SPL or SPL DSECT, you want the field names to begin with characters other than SPL. This avoids the occurrence of duplicate names if your program includes the macro two or more times; for example several times with TYPE=GEN and once with TYPE=MAP.

For xxx, specify the string of up to three characters with which you want the field names to begin.

PWD={password|(reg)}
The operand specifies the password associated with the queue entry to be retrieved or manipulated. The password must be alphameric and not longer than eight characters; if it is shorter, it is to be defined left justified and padded with blanks.

For a request with TYPE=GEN, specify a password (if this is feasible) as a character constant.

For a request with TYPE=UPD, specify the label of an eight-byte field that contains the password.

If you omit the operand, VSE/POWER defaults to a password of eight blanks. This does not give your program access to a queue entry submitted without password protection via a local spool device.

If you use register notation, the specified register must point to the eight-byte field that contains the password.

QUEUE={LST|PUN|RDR|XMT}

The operand specifies the queue that is to be accessed. The queue specifications valid for the various function requests are indicated by an X in the table below:

	QUEUE=			
Function Specification	LST	PUN	RDR	XMT
REQ=CTL	X	X	X	X
REQ=GET	X	X	X	
REQ=PUT: spooling job(s)			X	
REQ=PUT: spooling output	X	X		

If your program submits a job for processing at another node (of your computer system's network), then it must define this in the * $$ JOB statement for the job.

If your program submits output data for transmission to another node, then the target node's name and user-ID must be defined in the fields SPLDTNN and SPLDTUID, respectively, of the applicable SPL.

REQ={CTL|GET|PUT}

The operand specifies the type of function to be performed. The set of operands that applies to each of these basic function requests is given below. In the operand lists, M = mandatory and O = optional. Specify:

CTL To pass to VSE/POWER a control request or a command for execution. Operands that apply to a CTL request (where: M = mandatory; O = optional):

FUNC=function	M
JOBN={jobname|(reg)}	M See * below
QUEUE={RDR|LST|PUN|XMT}	M See * below
USERID={userid|(reg)}	M
CLASS={class|(reg)}	O
JNUM={fieldname|(reg)}	O
JSUF={fieldname|(reg)}	O
NEWVAL={field|(reg)}	O
OPT=FORMAT	O
PWD={password|(reg)}	O

* Optional if your program passes a command to VSE/POWER.

GET To retrieve, from the specified VSE/POWER queue, the named queue entry. Operands that apply to a GET request (where: M = mandatory; O = optional):

CLASS={class|(reg)}	M
JOBN={jobname|(reg)}	M
QUEUE={RDR|LST|PUN}	M
USERID={userid|(reg)}	M
JNUM={fieldname|(reg)}	O
JSUF={fieldname|(reg)}	O
MODE=BROWSE	O
MODE=GENERIC	O
OPT=([ALLCPY][,CTLREC][,NOWAIT][,RETSEP])	O
PWD={password|(reg)}	O

PUT To have job(s) spooled to VSE/POWER input queues (RDR, XMT) and Job output to the VSE/POWER output queues (LST, PUN, XMT).

Operands that apply to a PUT-job request (where: M = mandatory; O = optional):

QUEUE=RDR	M
USERID={userid|(reg)}	M
OPT=NOWAIT	O
PWD={password|(reg)}	O

Operands that apply to a PUT-output request (where: M = mandatory; O = optional):

QUEUE={LST|PUN}	M
JOBN={jobname|(reg)}	M
USERID={userid|(reg)}	M
CLASS={class|(reg)}	O
MODE={APPEND|RESTART}	O
OPT=NOWAIT	O
PWD={password|(reg)}	O

Spooling of output data may require that your program sets up a certain number of SPL fields individually. For more information about setting up SPL fields, see "Request a PUT Service – Submission of Output Data" on page 72.

Note: For spooling job(s) or output to the XMT queue, see the description of the QUEUE operand.

SPL={splname|(reg)}
The operand specifies the address of the spool parameter list (SPL) that is to be used. The operand applies only if your specify TYPE=UPD.

If you do not use register notation, then the code generated for your PWRSPL macro causes a pointer to the SPL to be loaded into register 1. Save this register's content before you issue the macro. If you code the macro with a name in the name field, that name must be identical with the symbolic address you specify for splname.

USERID={userid|(reg)}
The operand specifies the user identifier associated with the queue entry that is to be retrieved, submitted, or manipulated.

Note: ANY is not recommended as userid.

For a request with TYPE=GEN, specify the actual identifier as a character constant.

For a request with TYPE=UPD, specify the label of an eight-byte field that contains the identifier left justified and padded with blanks.

If you use register notation, the specified register must point to the eight-byte field that contains the identifier.

XPCC Macro

The macro initiates a cross-partition communication service.

Macro Format

Name	Operation	Operands

[name]	XPCC	XPCCB={address|(1)|(S,address)},
		FUNC={function|(reg-no.)}
		[,BUFFER={buffname|(reg-no.)|(S,addr)}]

Description of Operands

XPCCB={address|(1)|(S,address)},
The operand defines the address of the XPCCB, a control block containing request-related information. For more details about the block, see "XPCCB Macro" on page 118.

FUNC={function|(reg-no.)}
The operand defines the type of request. You can specify:

CONNECT To have the system provide a communication path to VSE/POWER. Your program can have two or more communication paths set up by using for each path a separate copy of the XPCCB you used for program identification (FUNC=IDENT).

DISCONN To have the system disconnect the currently used (and no longer required) communication path to VSE/POWER.

DISCPRG To have the system remove the existing communication path (set up by a FUNC=CONNECT) at once. This may interrupt the transfer of data from your program to VSE/POWER or vice versa.

For GET-service processing, VSE/POWER retains the affected queue entry with its disposition and priority unchanged.

For PUT-service processing, the interrupted submission has to be restarted either at a checkpoint or from the beginning.

IDENT To make your program known to the system as a spool-access support user. This is some kind of a "log-on" service.

SENDR To have VSE/POWER process the desired service.

TERMIN To finish using the spool-access support. This is some kind of a "log-off" service. Specify this operand if none of your program's tasks requires any further access to VSE/POWER services.

BUFFER={buffname|(reg-no.)|(S,addr)}
The operand may be used to define your program's send buffer. If you use the operand, the system ignores the area definition given by BUFFER=specification in the associated XPCCB macro.

Return Information

The system supplies return information in register 15 and in field IJBXRETC of the XPCCB; it may supply additional return information in field IJBXREAS. You should test this information along with testing the posting of IJBXSECB, the send-event control block.

VSE/POWER supplied return information in the XPCCB's user data area (field IJBXRUSR) is listed in the preceding sections that discuss CTL, GET, and PUT service requests.

Figure 39 on page 115 lists the mnemonics that you can use to test the return and reason codes supplied by the system. This list is followed by a short description of these mnemonics. The mnemonics are also listed and described in the DSECT generated by the assembler for the MAPXPCCB macro.

Reg.	Mnemonic in XPCCB Field		FUNC=					
15	IJBXRETC	IJBXREAS	CONNECT	DISCONN	DISPRG	IDENT	SENDR	TERMIN
00	IJBXREOK		X	X	X	X	X	X
04	IJBXAPSP					X		
	IJBXDAPP					X		
	IJBXNIDN		X					
	IJBXNCNN		X					
08	IJBXCBSY						X	
	IJBXNDC1			X				
	IJBXNDC2			X				
	IJBXNOC1						X	
	IJBXNOC2						X	
	IJBXNOC3						X	
	IJBXNOSY		X					
	IJBXNSTO		X					
	IJBXNTRM							X
	IJBXQSCE		X					
	IJBXTMCR		X		X			
	IJBXWCBA			X				
	IJBXWCBK		X	X	X	X	X	X
	IJBXWIDK		X					X
	IJBXWIND						X	
	IJBXWLST						X	
	IJBXWOWN			X	X		X	X
	IJBXWPID			X	X		X	
		IJBXCPRG					X	
		IJBXDISC					X	
		IJBXABDC					X	

Figure 39 (Part 1 of 3). Return and Reason Codes – XPCC Macro

The Codes in Field IJBXRETC (Return Codes)

Mnemonic	Equated Value	Explanation
IJBXREOK	00	Request completed successfully.
IJBXAPSP	02	Identification is requested with the same application from the same partition. Connect to VSE/POWER is possible.
IJBXCBSY	12	The communication path to be used is busy.
IJBXDAPP	01	Identification is requested with the same application from a different partition. Connect to VSE/POWER is possible.
IJBXNCNN	05	VSE/POWER has identified itself but not issued a CONNECT request (see "Note" below).
IJBXNDC1	15	The communication path is being used (a request from your program is being processed).
IJBXNDC2	16	The communication path is being used (a request issued by VSE/POWER is being processed).
IJBXNIDN	04	VSE/POWER has not yet identified itself to the system (see "Note" below).
IJBXNOC1	18	The communication path to be used does not exist.
IJBXNOC2	19	VSE/POWER came to a normal end of processing.
IJBXNOC3	1A	VSE/POWER came to an abnormal end of processing.
IJBXNOSY	0F	The name specified for TOAPPL in XPCCB is not SYSPWR.
IJBXNSTO	0E	No storage available for setting up the required control blocks.
IJBXQSCE	17	VSE/POWER is being shut down.
IJBXTMCR	0D	Too many CONNECT requests were issued by the requestor.

Note: Have your program wait for field IJBXCECB to be posted.

Figure 39 (Part 2 of 3). Return and Reason Codes – XPCC Macro

The Codes in Field IJBXRETC (Return Codes) - Continued

Mnemonic	Equated Value	Explanation
IJBXWCBA	1C	The request uses an XPCCB other than the one used with the FUNC=CONNECT request for setting up the communication path.
IJBXWCBK	06	The XPCCB has an invalid format.
IJBXWIDK	07	Wrong system-assigned cross-partition identifier.
IJBXWIND	0A	In the defined buffer list, at least one of the indicators is wrong.
IJBXWLST	0B	One of the following: - Too many buffers are specified. - The total length of the buffers exceeds 16M bytes. - One of the buffers in the list has a length of zero.
IJBXWOWN	09	The task that issued the request is not authorized to use the communication path.
IJBXWPID	08	Wrong system-assigned path identifier.

The Codes in Field IJBXREAS (Reason Codes)

Mnemonic	Equated Value	Explanation
IJBXDISC	40	VSE/POWER issued a disconnect request (see "Note" below).
IJBXABDC	80	VSE/POWER was disconnected as a result of an abnormal end (see "Note" below).

Note: The reason code, if it occurs, is inserted into field IJBXREAS by an OI instruction.

Figure 39 (Part 3 of 3). Return and Reason Codes - XPCC Macro

XPCCB Macro

The macro sets up a cross-partition control block. Logically, the block represents one communication path.

Format of the Macro

Name	Operation	Operands
[name]	XPCCB	APPL=name,TOAPPL={ANY|SYSPWR} [,BUFFER={buffname|(buffname,length)}] [,REPAREA=(areaname,length)]

Description of Operands

APPL=name

For name, specify the name of your program. The characters SYS as the first three characters of a name are reserved for IBM subsystems.

BUFFER={buffname|(buffname,length)}

In the operand, buffname is the name of your program's send buffer.

For length, specify the buffer's length in number of bytes. For the transfer of data to VSE/POWER, this buffer may have a length of up to 64K bytes.

If you do not specify a length, your program must insert this length into field IJBXBLN of the XPCCB.

REPAREA=(areaname,length)

In the operand, areaname is the name of your program's reply buffer.

For length, specify the buffer's length in number of bytes. For the transfer of data from VSE/POWER, this buffer may have a length of up to 64K bytes.

TOAPPL={ANY|SYSPWR}

Specify:

TOAPPL=ANY

If your application makes use of the external device support (for more details see Chapter 3).

TOAPPL=SYSPWR
If your application accesses VSE/POWER services for queue manipulation and for the retrieval or submission of jobs and output.

SPOOL-ACCESS SUPPORT PROGRAMMING EXAMPLE

This section consists of:

1. The set of statements that causes the source code of the spool-accesss support example to be assembled, linked, and executed.

2. An inline macro definition.

3. The source code, which is provided under "Programming Example Source Code" on page 123 primarily for study and reference purposes.

For ease of locating labels, a "Labels-to-Page Cross Reference" is provided below. Referenced page numbers are given in the upper right-hand corner on odd-numbered pages and in the upper left-hand corner on even-numbered pages.

Labels-to-Page Cross Reference

Label	Example Page
ABNPOW	22
BADREAS	22
BUFLN	23
BUFPTR	23
CONCT	5
CONNOK	5
CTLA1	9
CTLA2	9
CTLA3	10
DATACARD	25
DATAPTR	23
DATDSPLY	17
DISCT	19
DISP	23
DISPLAY	17
DSPL0	18
DSPL1	18
DSPL2	18
DSPL3	18
EIGHT	23
EIGHTDC	23
ENDIND	25
FAILM1	24
FAILM2	24
FAILM3	24

Label	Example Page
FAILM4	24
FAILFUNC	24
FAILLABL	24
FAILMSG	24
FAILREAS	24
FAILRETC	24
FAILPWFB	24
FAILPWRC	24
FILLBUF	7
FILLBUFO	15
FINDUMP	20
FINEND	20
FORMS	23
F1	24
F2	24
F3	24
F4	24
F5	24
F6	24
F7	24
GETB1	11
GETB2	12
GETB3	12
GETFCT	23
GQUIT	13
HELP	23

Label	Example Page
IDENT	4
INTECB	23
JCL1	25
JCL2	25
JECL1	25
JECL2	25
JECL3	25
JOBNAME	23
JOBNLAB	23
JOBNUM	23
KEEPRCT	16
LAB1	15
LASTPREC	25
LISTCECB	23
LISTEND	23
MSGDSPLY	20
MSGRCFB	20
MSGREAS	20
MSGRETC	20
M1	23
M7	23
NOOFRECS	23
NOTNLB	16
ONE	23
OWNSPL	26
OWNSPLDS	26
OWNXPCCB	25
POSTBIT	23
PRIOR	23
PUTA1	6
PUTA2	7
PUTA3	7
PUTA4	10
PUTB1	14
PUTB2	15
PUTB3	16
PWRRECNO	23
RA	26

Label	Example Page
RB	26
RC	26
RD	26
RE	26
REASOK	23
RECORDCT	23
REPLBUF	25
REQFAIL	19
RF	26
R0	26
R1	26
R2	26
R3	26
R4	26
R5	26
R6	26
R7	26
R8	26
R9	26
SAMPIN	3
SENDBUF	25
SDEOD	16
SDNDB	16
SEPPAGE	23
SENDR	21
SUCCM1	24
SUCCM2	24
SUCCM3	24
SUCCM4	24
SUCCM6	24
SUCCM7	24
SUCCM8	24
TERMCONN	22
TERMN	19
TERMQSCE	5
TESTRETC	22
TRTAB	23
WAITCECB	5
WAITLIST	23
WAITSECB	22
ZERO	23

Control Statements for Assembly, Link-Editing, and Execution

```
* $$ JOB JNM=PWRSARUN,CLASS=A,DISP=D
// JOB PWRSARUN
// OPTION CATAL
// LIBDEF *,SEARCH=(...)
*
* PROVIDE FOR (...) SEARCH CHAIN FOR VSE/POWER SUBLIBRARY
* AND VSE/AF SUPERVISOR MACRO SUBLIBRARY
*
// LIBDEF *,CATALOG=...
*
* PROVIDE FOR ... LINK SUBLIBRARY FOR PWRSASEX
*
// EXEC ASSEMBLY,SIZE=100K
         COPY PWRSASEX
         END
/*
// EXEC LNKEDT
// EXEC PWRSASEX
/&
* $$ EOJ
```

Inline Macro Definition

This macro definition, which precedes the source code, provides for a display of messages on the system console. Only the beginning and end of the instructions of this definition are shown here.

```
         TITLE 'PWRSASEX  -  SAS EXAMPLE PROGRAM'
         MACRO
&LABEL   DPLAY &LINE,&LENGTH,&ID=1
         GBLB  &FDSP(15)
         LCLA  &LINLEN,&LENLEN
         LCLC  &LENREG,&LINREG,&DISP
         LCLB  &LENSW,&LINSW,&TXT,&DEF
         AIF   (T'&ID EQ 'N' AND &ID LE 15).L001
         MNOTE 8,'ID NOT NUMERIC OR GREATER THAN 15'
         MEXIT
.L001    AIF   (T'&LINE NE 'O').L002
          ... ... ...
          ... ... ...
          ... ... ...
.L018    ANOP
         L     0,=A(&LINE)
.L019    ANOP
         STCM  0,7,DSCCW&ID+1
         L     1,=A(DSCCB&ID)
         EXCP  (1)
         WAIT  (1)
         MEND
```

Programming Example Source Code

```
                                                  Page 1 of 26

         PUNCH ' PHASE PWRSASEX,*'
***********************************************************************
**                                                                   **
**                      P W R S A S E X                              **
**                                                                   **
**       VSE/POWER SPOOL ACCESS SUPPORT:  EXAMPLE PROGRAM            **
**                                                                   **
***********************************************************************
*                                                                     *
*   THIS PROGRAM - NAMED PWRSASEX - ACTS AS A SPOOL-ACCESS-SERVICE    *
*   USER THAT INTERACTS WITH VSE/POWER VIA USING THE AVAILABLE        *
*   SPOOL-ACCESS SUPPORT.                                             *
*                                                                     *
*   PWRSASEX CAN RUN IN ANY PARTITION, UNDER OR OUTSIDE THE CONTROL   *
*   OF VSE/POWER.  FOR SUCCESSFUL COMPLETION, HOWEVER, AN ADDITIONAL  *
*   PARTITION UNDER CONTROL OF VSE/POWER AND WITH EXECUTION CLASS=A   *
*   MUST BE WAITING FOR WORK.                                         *
*                                                                     *
*   THE PROGRAM'S OPERATIONAL STEPS ARE:                              *
*                                                                     *
*   1.  IDENTIFY ITSELF TO THE SYSTEM'S XPCC SUPPORT WITH THE USER    *
*       IDENTIFICATION "PWRSASEX."                                    *
*                                                                     *
*   2.  TRY TO ESTABLISH A COMMUNICATION PATH TO VSE/POWER -- TERMIN- *
*       ATE IF THIS PATH CANNOT BE ESTABLISHED WITHIN TWO MINUTES     *
*                                                                     *
*   3.  USE THE PUT SERVICE TO SUBMIT THE JOB 'EXAMPLE' TO THE        *
*       VSE/POWER RDR QUEUE FOR EXECUTION IN CLASS=A.                 *
*                                                                     *
*   4.  USE THE CTL SERVICE TO SUBMIT A FIXED-FORMAT PDISPLAY COMMAND *
*       IN ORDER TO LOCATE THE OUTPUT OF JOB 'EXAMPLE' IN THE         *
*       VSE/POWER LST QUEUE, AND SHOW THE QUEUE-DISPLAY MESSAGE ON    *
*       THE CONSOLE.  IF THE OUTPUT IS NOT YET AVAILABLE, THE PROGRAM *
*       REISSUES THE PDISPLAY COMMAND EVERY 10TH OF A SECOND FOR TWO  *
*       MINUTES.  IF THEN THE OUTPUT IS STILL NOT AVAILABLE, PWRSASEX *
*       TERMINATES.                                                   *
*                                                                     *
*   5.  RETRIEVE THE LST QUEUE ENTRY 'EXAMPLE' USING THE GET SERVICE. *
*                                                                     *
*       THE PROGRAM CAUSES THE COMPLETE ENTRY TO BE DISPLAYED ON THE  *
*       CONSOLE.  PWRSASEX ISSUES A GET-RESTART REQUEST THAT POSITIONS*
*       THE RETRIEVAL POINTER IN THE MIDDLE OF THE QUEUE ENTRY AND    *
*       REDISPLAYS THE SECOND HALF.                                   *
*                                                                     *
*       PWRSASEX ENDS GET PROCESSING BY ISSUING A QUIT REQUEST.       *
*                                                                     *
***********************************************************************
```

Page 2 of 26

```
************************************************************************
*                                                                      *
*   6.  SUBMIT THE DATA CARDS OF JOB 'EXAMPLE' TO THE VSE/POWER        *
*       LST-QUEUE AS ENTRY 'EXAMPSEG' AND ISSUE PUT-SEGMENT REQUESTS   *
*       TO GET THREE SEGMENTS OF EQUAL SIZE.                           *
*                                                                      *
*       NOTE: THE ASA CONTROL 'CHARACTER PRINT-AND-SKIP-2' IS USED     *
*             FOR THE SUBMITTED LINES.                                 *
*                                                                      *
*   7.  DISCONNECT THE COMMUNICATION PATH TO VSE/POWER.                *
*                                                                      *
*   8.  TERMINATE (LOG OFF FROM) THE VSE XPCC SUPPORT.                 *
*                                                                      *
************************************************************************
*                                                                      *
*  THE FOLLOWING MACROS ARE REQUIRED:                                  *
*                                                                      *
*      SYSTEM MACROS:  XPCC                                            *
*                      XPCCB                                           *
*                      MAPXPCCB                                        *
*                                                                      *
*                      SETIME                                          *
*                      WAITM                                           *
*                      WAIT                                            *
*                                                                      *
*      VSE/POWER:      PWRSPL                                          *
*                                                                      *
*                                                                      *
*      AN INLINE MACRO (AVAILABLE WITH THE EXAMPLE).  IT IS USED FOR   *
*      DISPLAYING MESSAGES ON THE CONSOLE.  THE MACRO CALLS ARE IN     *
*      THE FORMAT:                                                     *
*                                                                      *
*          DPLAY MESSAGE-LABEL,LENGTH                                  *
*          DPLAY (REG1),(REG2)                                         *
*                                                                      *
*  NOTE: LINES WITH THE @-SIGN AT THE END REPRESENT THE INTERFACE     *
*        TO VSE/POWER.                                                 *
*        LINES WITHOUT THE @-SIGN AT THE END REPRESENT THE INTERFACE   *
*        TO THE SYSTEM'S XPCC SUPPORT (STEPS 1, 2, 7, AND 8).          *
*                                                                      *
************************************************************************
```

Page 3 of 26

```
         EJECT
         SPACE 2
*        REGISTER USAGE
         SPACE 2
*        R0 - **** - WORK REGISTER
*        R1 - **** - WORK REGISTER, ALSO USED BY PWRSPL MACRO
*        R2 - **** - WORK REGISTER
*        R3 - **** - WORK REGISTER
*        R4 - **** - ADDR REG FOR CROSS PARTITION CONTROL BLOCK XPCCB
*        R5 - **** - ADDRESS REGISTER FOR USER DATA TO BE SENT
*        R6 - **** - ADDRESS REGISTER FOR RECEIVED USER DATA
*        R7 - **** - ADDRESS REGISTER FOR SPL DSECT
*        R8 - **** - FIRST BASE REGISTER OF PWRSASEX
*        R9 - **** - SECOND BASE REGISTER OF PWRSASEX
*        RA - **** - WORK REGISTER
*        RB - **** - WORK REGISTER
*        RC - **** - WORK REGISTER
*        RD - **** - BRANCH AND LINK REGISTER FOR SENDR SUBROUTINE
*        RE - **** - BRANCH AND LINK REGISTER FOR DATDSPLY SUBROUTINE
*        RF - **** - MACRO CALL RETURN CODE REGISTER
         EJECT
SAMPIN   START 120                START OF THIS SAMPLE PROGRAM
         BALR  R8,0               GET START ADDRESS
         USING *,R8,R9            ESTABLISH ADDRESSABILITY
         SPACE 2
         LA    R9,4095(,R8)       LOAD SECOND BASE REGISTER WITH
         LA    R9,1(,R9)          CONTENTS OF FIRST + 4096
         SPACE 2
         LA    R4,OWNXPCCB        GET ADDR OF CROSS PART. CONTROL BLK
         USING IJBXPCCB,R4        ESTABLISH ADDRESSABILITY FOR DSECT
         SPACE 2
         LA    R5,IJBXSUSR        GET ADDR OF USER DATA TO BE SENT
         USING PXUUSER,R5         ESTABLISH ADDRESSABILITY FOR DSECT
         SPACE 2
         LA    R6,IJBXRUSR        GET ADDR OF RECEIVED USER DATA
         USING PXPUSER,R6         ESTABLISH ADDRESSABILITY FOR DSECT
         SPACE 2
         LA    R7,OWNSPL          GET ADDR OF SPL
         USING OWNSPLDS,R7        ESTABLISH ADDRESSABILITY FOR DSECT
         EJECT
```

Page 4 of 26

```
***********************************************************************
**          >> IDENTIFY PWRSASEX AS VSE/AF XPCC USER <<              **
** IF THE MACRO FAILS, THE PROGRAM DISPLAYS A MESSAGE AND TERMINATES:**
**  - WITHOUT A DUMP IF IT FAILED BECAUSE OF LACK OF STORAGE         **
**  - WITH A DUMP OTHERWISE.                                         **
***********************************************************************
         SPACE 1
IDENT    DS    0H
         SPACE 1
         XPCC  XPCCB=(R4),FUNC=IDENT   IDENTIFY 'PWRSASEX' TO AF-XPCC
         SPACE 1
         CLM   RF,M1,EIGHTDC       WAS RETURN CODE X'08' GIVEN BACK ?
         BNO   CONCT               ..NO, CONTINUE WITH CONNECTION
         SPACE 1
         MVC   FAILFUNC,=C'IDENTIFY'  INSERT FAILING FUNCTION INTO MSG
         BAL   RE,MSGRETC          INSERT XPCC RETURN CODE INTO MSG
         MVC   FAILLABL,=C'IDENT'  INSERT CODE LABEL FOR DIAGNOSTIC
         BAL   RE,MSGDSPLY         DISPLAY MESSAGE ON CONSOLE
         CLI   IJBXRETC,IJBXNSTO   DID IDENT FAIL DUE TO NO STORAGE ?
         BE    FINEND              ..YES, TERMINATE WITH EOJ MACRO
         B     FINDUMP             BRANCH TO TERMINATION WITH DUMP
         EJECT
```

Page 5 of 26

```
************************************************************************
**      >> ESTABLISH THE XPCC CONNECTION TO VSE/POWER <<              **
** IF THE MACRO FAILS, THE PROGRAM DISPLAYS A FAILURE MESSAGE AND     **
** TERMINATES.  THE PROGRAM WAITS UP TO TWO MINUTES FOR THE CONNEC-   **
** TION TO BE COMPLETED.                                              **
** IF THE CONNECTION IS ESTABLISHED AS REQUESTED, THE PROGRAM DIS-    **
** PLAYS A CONFIRMATION MESSAGE.                                      **
************************************************************************
         SPACE 1
CONCT    DS    0H
         SPACE 1
         XPCC  XPCCB=(R4),FUNC=CONNECT        CONNECT TO VSE/POWER
         SPACE 1
         LTR   RF,RF                IS CONNECTION ALREADY AVAILABLE ?
         BZ    CONNOK               ..YES, BYPASS WAIT FOR CONNECTION
         SPACE 1
         CLM   RF,M1,EIGHTDC         WAS RETURN CODE X'08' GIVEN BACK ?
         BL    WAITCECB             ..NO, MUST BE '04', SO WAIT FOR CECB
         CLI   IJBXRETC,IJBXQSCE    DID POWER GIVE XPCC TERMQSCE ?
         BE    TERMQSCE             ..YES, GO TO HANDLE THAT STATE
         MVC   FAILFUNC,=C'CONNECT '   INSERT FAILING FUNCTION INTO MSG
         BAL   RE,MSGRETC           INSERT XPCC RETURN CODE INTO MSG
         MVC   FAILLABL,=C'CONCT'   INSERT CODE LABEL FOR DIAGNOSTIC
         BAL   RE,MSGDSPLY          DISPLAY MESSAGE ON CONSOLE
         CLI   IJBXRETC,IJBXNSTO    DID CONNECT FAIL DUE TO NO STOR. ?
         BE    TERMN                ..YES, GO TO CLOSE XPCC INTERFACE
         SPACE 1
         B     FINDUMP              GO TO TERMINATION WITH DUMP
         SPACE 1
TERMQSCE DS    0H
         DPLAY FAILM1,72            DISPLAY FAILURE MESSAGE
         SPACE 1
         B     TERMN                GO TO CLOSE XPCC INTERFACE CORRECTLY
         SPACE 1
WAITCECB DS    0H                   CONNECTION IS STILL 'PENDING'
         SETIME 120,INTECB          INSTALL WAIT INTERVAL OF TWO MIN.
         LA    R3,IJBXCECB          LOAD ADDRESS OF CONNECTION ECB
         ST    R3,LISTCECB          COMPLETE WAITLIST
         WAITM WAITLIST             WAIT FOR CONNECTION OR 2 MIN. COMPL.
         TM    IJBXCECB+2,POSTBIT   CONNECTION COMPLETE?
         BO    CONNOK               ...YES, CONTINUE AT CONNOK
         SPACE 1
         DPLAY FAILM3,72            ISSUE MSG THAT TIME LIMIT EXCEEDED
         SPACE 1
         B     DISCT                GO TO DISCONNECT AND TERMINATE
         SPACE 1
CONNOK   DS    0H                   NOW, CONNECTION ECB IS POSTED
         DPLAY SUCCM1,72
         EJECT
```

Page 6 of 26

```
**********************************************************************@
**      >>          PUT-REQUEST TO RDR QUEUE          <<             *@
**  THE JOB 'EXAMPLE' IS SUBMITTED TO THE VSE/POWER RDR QUEUE.       *@
**********************************************************************@
         SPACE 1                                                      @
*        REGISTER USAGE FOR PUT-REQUEST TO RDR QUEUE                  @
         SPACE 2                                                      @
*        R3 -  *****   - WORK REGISTER                                @
*        RA - BUFPTR   - POINTER FOR THE SEND BUFFER                  @
*        RB - DATAPTR  - POINTER FOR THE INPUT CARDS                  @
*        RC -  *****   - TEMPORARY ADDR. REG FOR SPL DSECT            @
         SPACE 2                                                      @
*    THE GENERATED SPL (OWNSPL) IS UPDATED INDICATING A PUT OPEN      @
*    REQUEST AND THEN SENT TO VSE/POWER.                              @
         SPACE 2                                                      @
PUTA1    DS    0H                                                     @
         PWRSPL TYPE=UPD,SPL=OWNSPL,REQ=PUT,QUEUE=RDR                 @
         SPACE 2                                                      @
         MVI   PXUBTYP,PXUBTSPL    INDICATE BUFFER TYPE = SPL         @
         MVI   PXUACT1,0           CLEAR ALL OTHER BYTES IN PXUUSER,  @
         MVI   PXUSIGNL,0          WHICH MAY BE CHANGED BY THE USER   @
         SPACE 1                                                      @
*        THE SPL IS DIRECTLY USED AS XPCC SEND BUFFER                 @
         SPACE 1                                                      @
         STCM  R7,M7,IJBXADR       INSERT SPL ADDRESS AS BUFFER ADDR. @
         LA    R3,SPLGLEN          LOAD LENGTH OF SPL                 @
         ST    R3,IJBXBLN          INSERT BUFFER LENGTH INTO XPCCB    @
         SPACE 1                                                      @
         MVC   FAILLABL,=C'PUTA1'  INSERT CODE LABEL FOR DIAGNOSTIC   @
         BAL   RD,SENDR            GO TO SENDR ROUTINE                @
         CLI   PXPRETCD,PXPRCOK    WAS POWER RETURN CODE ZERO?        @
         BNE   REQFAIL             NO, GO TO HANDLE REQUEST FAILURE   @
*    THE EXTENDED SPL RETURNED BY VSE/POWER IS IGNORED                @
         SPACE 2                                                      @
*    THE SEND BUFFER IS FILLED WITH INPUT CARDS (EACH CARD            @
*    PRECEDED BY A RECORD PREFIX) UNTIL NO MORE CARD FITS.            @
*    THE BUFFER IS THEN PASSED TO VSE/POWER IN THE ACTUALLY           @
*    USED LENGTH.                                                     @
         SPACE 1                                                      @
         MVI   PXUBTYP,PXUBTNDB    BUFFER TYPE = NORMAL DATA BUFFER   @
         MVI   PXUACT1,0           CLEAR ACTION BYTE                  @
         SPACE 1                                                      @
         LA    BUFPTR,SENDBUF      GET ADDRESS OF SEND BUFFER         @
         STCM  BUFPTR,M7,IJBXADR   INSERT BUFFER ADDRESS INTO XPCCB   @
         LA    DATAPTR,JECL1       GET ADDR OF FIRST INPUT CARD       @
         SPACE 1                                                      @
```

Page 7 of 26

```
FILLBUF  DS    0H                                                     @
         CLC   ENDIND,0(DATAPTR)    END OF FILE REACHED?              @
         BE    PUTA3               YES, GO TO SEND FINAL BUFFER       @
         CL    BUFPTR,LASTPREC     ENOUGH SPACE FOR ONE MORE RECORD?  @
         BH    PUTA2               NO, GO TO SEND NORMAL BUFFER       @
         USING RECPRFIX,BUFPTR     GET DSECT FOR RECORD LAYOUT        @
         XC    0(RECPRFXL,BUFPTR),0(BUFPTR)   CLEAR BYTES FOR PREFIX  @
         MVI   RECTYPE,RECTNORM    INSERT REC. TYPE IN REC. PREFIX    @
         LA    R3,L'DATACARD       LOAD LENGTH OF DATA CARD           @
         STH   R3,RECLNGTH         INSERT LENGTH OF DATA CARD IN PREF.@
         LA    BUFPTR,RECPRFXL(,BUFPTR)         SKIP PREFIX IN BUFFER  @
         DROP  BUFPTR                                                 @
         MVC   0(L'DATACARD,BUFPTR),0(DATAPTR) MOVE DATA INTO BUFFER  @
         LA    BUFPTR,L'DATACARD(,BUFPTR)   POINT TO NEXT FREE B.SPACE@
         LA    DATAPTR,L'DATACARD(,DATAPTR) POINT TO NEXT INPUT CARD  @
         B     FILLBUF             TRY TO FILL IN NEXT INPUT CARD     @
         SPACE 1                                                      @
PUTA2    DS    0H                                                     @
         LA    R3,SENDBUF          GET AGAIN START ADDR OF SEND BUFFER@
         SR    BUFPTR,R3           CALC. ACTUALLY USED BUFFER LENGTH  @
         ST    BUFPTR,IJBXBLN      INSERT ACTUAL BUF.LENGTH INTO XPCCB@
         MVC   FAILLABL,=C'PUTA2'  INSERT CODE LABEL FOR DIAGNOSTIC   @
         BAL   RD,SENDR            GO TO SENDR ROUTINE                @
         CLI   PXPRETCD,PXPRCOK    WAS POWER RETURN CODE ZERO?        @
         BNE   REQFAIL             NO, GO TO HANDLE REQUEST FAILURE   @
         LA    BUFPTR,SENDBUF      GET AGAIN ADDRESS OF SEND BUFFER   @
         B     FILLBUF             GO TO FILL BUFFER AGAIN            @
         SPACE 1                                                      @
*   THE BUFFER BEING FILLED WHEN END OF FILE WAS DETECTED             @
*   IS PASSED TO VSE/POWER AS FINAL BUFFER WITH AN END OF DATA        @
*   INDICATION IN THE USER DATA (=PUT CLOSE REQUEST).                 @
         SPACE 1                                                      @
PUTA3    DS    0H                                                     @
         SPACE 1                                                      @
         MVI   PXUACT1,PXUATEOD    INDICATE END OF DATA               @
         LA    R3,SENDBUF          GET AGAIN START ADDR OF SEND BUFFER@
         SR    BUFPTR,R3           CALC. ACTUALLY USED BUFFER LENGTH  @
         ST    BUFPTR,IJBXBLN      INSERT ACTUAL BUF.LENGTH INTO XPCCB@
         MVC   FAILLABL,=C'PUTA3'  INSERT CODE LABEL FOR DIAGNOSTIC   @
         BAL   RD,SENDR            GO TO SENDR ROUTINE                @
         CLI   PXPRETCD,PXPRCOK    WAS POWER RETURN CODE ZERO?        @
         BNE   REQFAIL             NO, GO TO HANDLE REQUEST FAILURE   @
         CLI   PXPFBKCD,PXP000K    WAS POWER FEEDBACKCODE ALSO ZERO?  @
         BNE   REQFAIL             NO, GO TO HANDLE REQUEST FAILURE   @
         SPACE 1                                                      @
```

Page 8 of 26

```
*   THE EXTENDED SPL RETURNED BY VSE/POWER IS ANALYZED.               @
*   JOBNAME AND JOBNUMBER ARE SAVED.                                  @
*   IF MESSAGES ARE QUEUED, A RETURN MESSAGE REQUEST IS SENT. SUB-    @
*   SEQUENTLY, THE DATDSPLY ROUTINE IS CALLED IN ORDER TO DISPLAY     @
*   THE RETURNED MESSAGES.                                            @
         SPACE 1                                                      @
         LA    RC,REPLBUF           GET AD. OF REPLY AREA FOR SPL DSECT@
         USING OWNSPLDS,RC          ESTABLISH ADDRESSABILITY FOR DSECT @
         MVC   JOBNAME,SPLGJB       SAVE JOBNAME RETURNED BY POWER     @
         MVC   JOBNUM,SPLGJN        SAVE JOBNUMBER RETURNED BY POWER   @
         DROP  RC                                                     @
         USING OWNSPLDS,R7          REESTABLISH ADDRESSABILITY FOR SPL @
         SPACE 1                                                      @
         TM    PXPINFO,PXPIMSG      ARE MESSAGES QUEUED?               @
         BNO   CTLA1                NO, CONTINUE WITH CONTROL REQUEST  @
         SPACE 1                                                      @
PUTA4    DS    0H                                                     @
         XC    IJBXBLN,IJBXBLN      INDICATE ZERO BUFFER LENGTH        @
         MVI   PXUBTYP,0            CLEAR BUFFER TYPE BYTE IN USER DATA@
         MVI   PXUACT1,PXUATRMR     INDICATE RETURN MESSAGE REQUEST    @
         MVC   FAILLABL,=C'PUTA4'   INSERT CODE LABEL FOR DIAGNOSTIC   @
         BAL   RD,SENDR             GO TO SENDR ROUTINE                @
         CLI   PXPRETCD,PXPRCOK     WAS POWER RETURN CODE ZERO?        @
         BNE   REQFAIL              NO, GO TO HANDLE REQUEST FAILURE   @
         BAL   RE,DATDSPLY          YES, GO TO DISPLAY RETURNED MSG-S  @
         EJECT                                                        @
```

Page 9 of 26

```
**********************************************************************@
**       >>         CONTROL REQUEST                   <<             *@
** A FIXED FORMAT PDISPLAY COMMAND IS SUBMITTED IN ORDER TO LOCATE   *@
** THE OUTPUT OF JOB 'EXAMPLE' IN THE LST QUEUE AND DISPLAY THE      *@
** ENTRY ON THE CONSOLE.                                             *@
**********************************************************************@
         SPACE 1                                                      @
*        REGISTER USAGE FOR CTL-REQUEST                               @
         SPACE 2                                                      @
*        RA - *****  - COUNTER FOR NUMBER OF WAIT INTERVALS           @
         SPACE 2                                                      @
*        THE SPL IS NOW UPDATED INDICATING A CTL REQUEST.             @
         SPACE 1                                                      @
CTLA1    DS    0H                                                     @
         PWRSPL TYPE=UPD,SPL=OWNSPL,QUEUE=LST,REQ=CTL,CLASS=S,          *
               JOBN=JOBNAME,JNUM=JOBNUM,FUNC=DISPLAY                  @
         SPACE 1                                                      @
         LA    RA,12               PREPARE COUNTER FOR WAIT INTERVALS @
         MVI   PXUBTYP,PXUBTSPL    INDICATE BUFFER TYPE = SPL         @
         MVI   PXUACT1,0           CLEAR ACTION BYTE                  @
         SPACE 1                                                      @
*        THE UPDATED SPL IS DIRECTLY USED AS XPCC BUFFER.             @
         SPACE 1                                                      @
         STCM  R7,M7,IJBXADR       INSERT SPL ADDRESS AS BUFFER ADDR. @
         SPACE 1                                                      @
*   FOR A CTL REQUEST IT IS NOT NECESSARY TO PASS THE WHOLE SPL       @
*   TO VSE/POWER. THEREFORE THE SHORT VERSION IS USED.                @
         SPACE 1                                                      @
         LA    R3,SPLGSLEN         LOAD LENGTH OF SPL (SHORT VERSION) @
         ST    R3,IJBXBLN          INSERT BUFFER LENGTH INTO XPCCB    @
         SPACE 1                                                      @
         MVC   FAILLABL,=C'CTLA2'  INSERT CODE LABEL FOR DIAGNOSTIC   @
CTLA2    DS    0H                                                     @
         BAL   RD,SENDR            GO TO SENDR ROUTINE                @
         CLI   PXPRETCD,PXPRCOK    WAS POWER RETURN CODE ZERO?        @
         BE    CTLA3               YES, CONTINUE WITH MSG DISPLAY     @
         SPACE 1                                                      @
```

Page 10 of 26

```
*   THE PROGRAM TESTS THE POWER RC/FBKCD TO SEE IF THE OUTPUT OF THE  @
*   JOB 'EXAMPLE' COULD BE LOCATED.                                    @
*   IF THIS OUTPUT COULD NOT YET BE LOCATED, THE PROGRAM REPEATS THE   @
*   CTL REQUEST EVERY 10 SECONDS IN ORDER TO WAIT FOR REQUEST COMPLE-  @
*   TION.  HOWEVER PWRSASEX DISCONNECTS AFTER 12 UNSUCCESSFUL AT-      @
*   TEMPTS.                                                            @
*   ANY OTHER RC/FBKCD COMBINATION SHOULD NOT OCCUR AND INDICATES A    @
*   FAILURE OF THE REQUEST.                                            @
         SPACE 1                                                       @
         CLI   PXPRETCD,PXPRCOKF   WAS POWER RETURN CODE X'04'         @
         BNE   REQFAIL             NO, GO TO HANDLE REQUEST FAILURE    @
         CLI   PXPFBKCD,PXP04DNF   WAS JOB NOT FOUND (=NOT YET COMPL.)@
         BNE   REQFAIL             NO, GO TO HANDLE REQUEST FAILURE    @
         SPACE 1                                                       @
         SETIME 10,INTECB          INSTALL WAIT INTERVAL OF 10 SEC.    @
         WAIT   INTECB             WAIT                                @
         BCT   RA,CTLA2            LOOP (MAX. 12 TIMES)                @
         SPACE 1                                                       @
         DPLAY FAILM4,72           DISPLAY FAILURE MESSAGE             @
         SPACE 1                                                       @
         B     DISCT               DISCONN AND TERMIN XPCC LINK, EOJ   @
         SPACE 1                                                       @
CTLA3    DS    0H                                                      @
         BAL   RE,DATDSPLY         GO TO DISPLAY RETURNED QUEUE ENTRY  @
         EJECT                                                         @
```

Page 11 of 26

```
**********************************************************************@
**            >>   GET REQUEST FROM LST QUEUE    <<                  *@
**  THE GET SERVICE IS USED TO RETRIEVE THE LST QUEUE ENTRY OF JOB   *@
**  'EXAMPLE' AND TO DISPLAY THE ENTRY ON THE CONSOLE.               *@
**  SUBSEQUENTLY, THE RESTART FUNCTION IS USED TO DISPLAY THE SECOND *@
**  HALF OF THE ENTRY AGAIN.                                         *@
**********************************************************************@
         SPACE 1                                                      @
*        REGISTER USAGE FOR GET-REQUEST FROM LST QUEUE                @
         SPACE 2                                                      @
*        R3 - ****    - WORK REGISTER                                 @
*        RA - BUFPTR  - POINTER FOR THE SEND BUFFER                   @
         SPACE 2                                                      @
*   ONLY PARAMETERS WHICH ARE DIFFERENT FROM THOSE USED IN THE        @
*   PREVIOUS CTL-REQUEST ARE SPECIFIED IN THE UPDATE SPL.             @
         SPACE 1                                                      @
         DPLAY SUCCM7,72           DISPLAY MESSAGE                    @
         SPACE 1                                                      @
GETB1    DS    0H                                                     @
         PWRSPL TYPE=UPD,SPL=(R7),REQ=GET                             @
         SPACE 1                                                      @
         MVI   PXUBTYP,PXUBTSPL    INDICATE BUFFER TYPE = SPL         @
         MVI   PXUACT1,0           CLEAR ACTION BYTE 1                @
         SPACE 1                                                      @
         STCM  R7,M7,IJBXADR       INSERT SPL ADDRESS AS BUFFER ADDR. @
         LA    R3,SPLGSLEN         LOAD LENGTH OF SPL (SHORT VERSION) @
         ST    R3,IJBXBLN          INSERT BUFFER LENGTH INTO XPCCB    @
         SPACE 1                                                      @
         MVC   FAILLABL,=C'GETB1'  INSERT CODE LABEL FOR DIAGNOSTIC   @
         BAL   RD,SENDR            GO TO SENDR ROUTINE                @
         CLI   PXPRETCD,PXPRCOK    WAS POWER RETURN CODE ZERO?        @
         BNE   REQFAIL             NO, GO TO HANDLE REQUEST FAILURE   @
         SPACE 1                                                      @
```

Page 12 of 26

```
*   THE VERIFICATION SPL RETURNED BY VSE/POWER, WHICH COULD BE CHECKED@
*   FOR USEFUL INFORMATION (SUCH AS FORMSID), IS IGNORED BY PWRSASEX. @
*   THEREFORE, A NULL BUFFER WITH THE 'SEND DATA' REQUEST CAN BE      @
*   PASSED TO VSE/POWER IMMEDIATELY.                                  @
         SPACE 1                                                      @
GETB2    DS    OH                                                     @
         XC    IJBXBLN,IJBXBLN     INDICATE ZERO BUFFER LENGTH        @
         MVI   PXUBTYP,0           CLEAR BUFFER TYPE BYTE IN USER DATA@
         MVI   PXUACT1,PXUATSDR    INDICATE SEND DATA REQUEST         @
         MVC   FAILLABL,=C'GETB2'  INSERT CODE LABEL FOR DIAGNOSTIC   @
         BAL   RD,SENDR            GO TO SENDR ROUTINE                @
         CLI   PXPRETCD,PXPRCOK    WAS POWER RETURN CODE ZERO?        @
         BNE   REQFAIL             NO, GO TO HANDLE REQUEST FAILURE   @
         MVI   GETFCT,C'G'         INDICATE: DATDSPLY IS CALLED BY GET@
         BAL   RE,DATDSPLY         GO TO DISPLAY RETURNED DATA        @
*                                  AND DO NOT RETURN UNTIL LAST DATA  @
*                                  RECORD IS DISPLAYED                @
         MVI   GETFCT,C' '         RESET INDICATION                   @
         SPACE 1                                                      @
         DPLAY SUCCM2,72           DISPLAY MSG TO INDICATE RESTART RQ.@
         SPACE 1                                                      @
*   A RESTART CONTROL RECORD IS BUILT IN THE SEND BUFFER AND          @
*   PASSED TO VSE/POWER. THE LOGICAL RECORD NUMBER - PREVIOUSLY       @
*   SAVED IN THE DATDSPLY ROUTINE - IS USED AS RESTART POINT.         @
         SPACE 1                                                      @
GETB3    DS    OH                                                     @
         SPACE 1                                                      @
         MVI   PXUBTYP,PXUBTCTL    BUFFER TYPE = CONTROL RECORD       @
         MVI   PXUACT1,0           CLEAR ACTION BYTE 1                @
         LA    BUFPTR,SENDBUF      GET ADDRESS OF SEND BUFFER         @
         STCM  BUFPTR,M7,IJBXADR   INSERT BUFFER ADDRESS INTO XPCCB   @
         SPACE 1                                                      @
         USING PXRSDSCT,BUFPTR     GET DSECT FOR RESTART CONTROL REC. @
         XC    O(PXRSLENG,BUFPTR),O(BUFPTR)   CLEAR RESTART CONTROL R.@
         MVI   PXRSTYPE,PXRSTRST   INDICATE RECORD TYPE = RESTART CTL.@
         MVC   PXRSRECN,PWRRECNO   INSERT PREVIOUSLY SAVED LOG. REC.# @
         LA    R3,PXRSLENG         LOAD LENGTH OF RESTART CTL. REC.   @
         STH   R3,PXRSRLEN         INSERT LENGTH INTO RESTART CTL. REC@
         ST    R3,IJBXBLN          INSERT LENGTH INTO XPCCB           @
         DROP  BUFPTR                                                 @
```

Page 13 of 26

```
         SPACE 1                                                   @
         MVC   FAILLABL,=C'GETB3'  INSERT CODE LABEL FOR DIAGNOSTIC   @
         BAL   RD,SENDR            GO TO SENDR ROUTINE                @
         CLI   PXPRETCD,PXPRCOK    WAS POWER RETURN CODE ZERO?        @
         BNE   REQFAIL             NO, GO TO HANDLE REQUEST FAILURE   @
         MVI   GETFCT,C'G'         INDICATE: DATDSPLY IS CALLED BY GET@
         BAL   RE,DATDSPLY         YES, GO TO DISPLAY RETURNED DATA   @
*                                  AND DO NOT RETURN UNTIL LAST DATA  @
*                                  RECORD IS DISPLAYED                @
         MVI   GETFCT,C' '         RESET INDICATION                   @
         SPACE 1                                                      @
*    A ZERO BUFFER INDICATING A QUIT REQUEST IS NOW SENT TO           @
*    VSE/POWER                                                        @
         SPACE 1                                                      @
GQUIT    DS    0H                                                     @
         XC    IJBXBLN,IJBXBLN     INSERT ZERO BUFFER LENGTH          @
         MVI   PXUBTYP,0           CLEAR BUFFER TYPE BYTE IN USER DATA@
         MVI   PXUACT1,PXUATABR    INDICATE QUIT REQUEST              @
         SPACE 1                                                      @
*    IF A CLOSE OR PURGE REQUEST IS DESIRED, ONE OF THE FOLLOWING     @
*    STATEMENTS MUST BE CODED INSTEAD OF THE PREVIOUS ONE             @
         SPACE 1                                                      @
*GCLOSE  MVI   PXUACT1,PXUATRQS    REQUIRED SETTING FOR A CLOSE REQU. @
*GPURGE  MVI   PXUACT1,PXUATPRG    REQUIRED SETTING FOR A PURGE REQU. @
         SPACE 1                                                      @
         MVC   FAILLABL,=C'GQUIT'  INSERT CODE LABEL FOR DIAGNOSTIC   @
         BAL   RD,SENDR            GO TO SENDR ROUTINE                @
         CLI   PXPRETCD,PXPRCOK    WAS POWER RETURN CODE ZERO?        @
         BNE   REQFAIL             NO, GO TO HANDLE REQUEST FAILURE   @
         SPACE 2                                                      @
         EJECT                                                        @
```

Page 14 of 26

```
**********************************************************************@
**          >>      PUT REQUEST TO LST QUEUE          <<             *@
**  THE DATA CARDS OF THE EXAMPLE JOB ARE SUBMITTED TO THE VSE/POWER *@
**  LST QUEUE AS 'EXAMPSEG'.  A SEGMENT REQUEST IS ISSUED            *@
**  AFTER EACH SEVENTH RECORD.                                       *@
**********************************************************************@
         SPACE 1                                                      @
*        REGISTER USAGE FOR PUT-REQUEST TO LST QUEUE                  @
         SPACE 2                                                      @
*        R2 - RECORDCT - RECORD COUNTER FOR SEGMENTATION IN LOOP      @
*        RA - BUFPTR   - POINTER FOR THE SEND BUFFER                  @
*        RB - DATAPTR  - POINTER FOR THE INPUT CARDS                  @
         SPACE 2                                                      @
*   AN UPDATED SPL IS SENT TO VSE/POWER INDICATING A PUT OPEN REQUEST @
*   FOR OUTPUT.                                                       @
         SPACE 2                                                      @
         DPLAY SUCCM8,72           DISPLAY MESSAGE                    @
         SPACE 1                                                      @
PUTB1    DS    OH                                                     @
         PWRSPL TYPE=UPD,REQ=PUT,SPL=OWNSPL,CLASS=Z,JOBN=JOBLAB,       *
               QUEUE=LST,MODE=RESET                                   @
         SPACE 2                                                      @
*   SET OUTPUT SPECIFIC FIELDS IN THE SPL                             @
         MVI   SPLDDP,DISP         INDICATE OUTPUT DISPOSITION        @
         MVI   SPLONSEP,SEPPAGE    INDICATE OUTPUT SEPARATOR PAGES    @
         MVI   SPLDPR,PRIOR        INDICATE OUTPUT PRIORITY           @
         MVC   SPLOFORM,FORMS      INDICATE OUTPUT FORMS              @
         MVI   SPLORCFM,SPLORASA   INDICATE ASA CC FOR OUTPUT         @
         SPACE 1                                                      @
         MVI   PXUACT1,0           CLEAR ACTION BYTE 1 IN USER DATA   @
         MVI   PXUBTYP,PXUBTSPL    INDICATE BUFFER TYPE = SPL         @
         SPACE 1                                                      @
         STCM  R7,M7,IJBXADR       INSERT SPL ADDRESS AS BUFFER ADDR. @
         LA    R3,SPLGLEN          LOAD LENGTH OF SPL                 @
         ST    R3,IJBXBLN          INSERT BUFFER LENGTH INTO XPCCB    @
         SPACE 1                                                      @
         MVC   FAILLABL,=C'PUTB1'  INSERT CODE LABEL FOR DIAGNOSTIC   @
         BAL   RD,SENDR            GO TO SENDR ROUTINE                @
         CLI   PXPRETCD,PXPRCOK    WAS POWER RETURN CODE ZERO?        @
         BNE   REQFAIL             NO, GO TO HANDLE REQUEST FAILURE   @
         SPACE 2                                                      @
```

Page 15 of 26

```
*   THE EXTENDED SPL RETURNED BY VSE/POWER IS IGNORED.                @
*   THE SEND BUFFER IS FILLED WITH INPUT CARDS, AND EACH CARD IS      @
*   PRECEDED BY A RECORD PREFIX.                                      @
*   THE OUTPUT IS ALWAYS SEGMENTED AFTER SEVEN CARDS.                 @
         SPACE 1                                                      @
PUTB2    DS    0H                                                     @
         MVI   PXUBTYP,PXUBTNDB    BUFFER TYPE = NORMAL DATA BUFFER   @
         MVI   PXUACT1,0           CLEAR ACTION BYTE 1 IN USER DATA   @
         SPACE 1                                                      @
         LA    BUFPTR,SENDBUF      GET ADDRESS OF SEND BUFFER         @
         STCM  BUFPTR,M7,IJBXADR   INSERT BUFFER ADDRESS INTO XPCCB   @
         LA    DATAPTR,DATACARD    GET ADDR OF FIRST INPUT CARD       @
         SPACE 1                                                      @
         LA    RECORDCT,NOOFRECS   INITIALIZE RECORD COUNTER          @
FILLBUFO DS    0H                                                     @
         CLC   JCL2(3),0(DATAPTR) END OF DATA REACHED?                @
         BE    SDEOD               YES, GO TO SEND FINAL BUFFER       @
         SPACE 1                                                      @
         CL    BUFPTR,LASTPREC     ENOUGH SPACE FOR ONE MORE RECORD?  @
         BH    SDNDB               NO, GO TO SEND NORMAL BUFFER       @
         SPACE 1                                                      @
         USING RECPRFIX,BUFPTR     GET DSECT FOR RECORD LAYOUT        @
         XC    0(RECPRFXL,BUFPTR),0(BUFPTR)   CLEAR BYTES FOR PREFIX  @
         MVI   RECTYPE,RECTNORM    INSERT REC. TYPE INTO REC. PREFIX  @
         MVI   RECCCODE,C'-'       SET ASA CC IN REC. PREFIX TO SKIP2 @
         LA    R3,NOOFRECS         MAX NUMBER OF RECORDS IN A SEGMENT @
         CLR   RECORDCT,R3         FIRST RECORD OF SEGMENT?           @
         BNE   LAB1                NO, CONTINUE AT LABEL LABL1        @
         MVI   RECCCODE,C'1'       SET ASA CC IN REC.PREF. TO NXT.PAGE@
         SPACE 1                                                      @
LAB1     DS    0H                                                     @
         LA    R3,L'DATACARD       LOAD LENGTH OF DATA                @
         STH   R3,RECLNGTH         INSERT LENGTH OF DATACARD IN PREFIX@
         LA    BUFPTR,RECPRFXL(,BUFPTR) SKIP PREFIX IN BUFFER         @
         MVC   0(L'DATACARD,BUFPTR),0(DATAPTR) MOVE DATA IN BUFFER    @
         LA    BUFPTR,L'DATACARD(,BUFPTR) POINT TO NEXT FREE BUFSPACE @
         LA    DATAPTR,L'DATACARD(,DATAPTR) POINT TO NEXT INPUT CARD  @
         BCT   RECORDCT,FILLBUFO . DO LOOP AND DECREMENT RECORDCOUNTER@
         SPACE 1                                                      @
         CLC   JCL2(3),0(DATAPTR) END OF DATA REACHED?                @
         BE    SDEOD               YES, GO TO SEND FINAL BUFFER       @
         MVI   PXUACT1,PXUATSGM    INDICATE OUTPUT SEGMENTATION       @
```

Page 16 of 26

```
SDNDB    DS    0H                                                    @
         LA    R3,SENDBUF          GET AGAIN START ADDR. OF SEND BUFF @
         SR    BUFPTR,R3           CALC. ACTUALLY USED BUFFER LENGTH  @
         ST    BUFPTR,IJBXBLN      INSERT ACTUAL BUF.LENGTH INTO XPCCB@
         MVC   FAILLABL,=C'SDNDB'  INSERT PART OF CODE LABEL FOR DIAGN@
         BAL   RD,SENDR            GO TO SENDR ROUTINE                @
         CLI   PXPRETCD,PXPRCOK    WAS POWER RETURN CODE ZERO?        @
         BNE   REQFAIL             NO, GO TO HANDLE REQUEST FAILURE   @
         LA    BUFPTR,SENDBUF      GET ADDRESS OF SEND BUFFER         @
         LTR   RECORDCT,RECORDCT   IS RECORD COUNTER ZERO?            @
         BNZ   KEEPRCT             NO, KEEP ACTUAL VALUE OF RECORDCT  @
         LA    RECORDCT,NOOFRECS   INITIALIZE REC COUNTER AGAIN       @
KEEPRCT  DS    0H                                                    @
         MVI   PXUACT1,0           CLEAR ACTION BYTE 1 IN USER DATA   @
         B     FILLBUFO            GOTO CHECK NEXT INPUT CARD         @
         SPACE 2                                                     @
SDEOD    DS    0H                                                    @
         MVI   PXUACT1,0           CLEAR ACTION BYTE 1 IN USER DATA   @
         MVI   PXUACT1,PXUATEOD    ACTION BYTE = END OF DATA          @
         LA    R3,SENDBUF          GET AGAIN START ADDR. OF SEND BUFF @
         SR    BUFPTR,R3           CALC. ACTUALLY USED BUFFER LENGTH  @
         ST    BUFPTR,IJBXBLN      INSERT ACTUAL BUF.LENGTH INTO XPCCB@
         L     R3,IJBXBLN          LOAD ACTUAL SEND BUFFER LENGTH     @
         LTR   R3,R3               IS BUFFER LENGTH  ZERO?            @
         BNZ   NOTNLB              NO, IND. NORMAL DATA BUFFER        @
         MVI   PXUBTYP,0           CLEAR BUFFER TYPE IN USER DATA     @
NOTNLB   DS    0H                                                    @
         MVC   FAILLABL,=C'SDEOD'  INSERT PART OF CODE LABEL FOR DIAGN@
         BAL   RD,SENDR            GO TO SENDR ROUTINE                @
         CLI   PXPRETCD,PXPRCOK    WAS POWER RETURN CODE ZERO?        @
         BNE   REQFAIL             NO, GO TO HANDLE REQUEST FAILURE   @
         SPACE 2                                                     @
*   THE EXTENDED SPL RETURNED BY VSE/POWER IS IGNORED.               @
         SPACE 1                                                     @
PUTB3    DS    0H                                                    @
         TM    PXPINFO,PXPIMSG     MESSAGES QUEUED?                   @
         BNO   DISPLAY             NO, GO TO DISPLAY INFO MESSAGES    @
         MVC   FAILLABL,=C'PUTB3'  INSERT CODE LABEL FOR DIAGNOSTIC   @
         XC    IJBXBLN,IJBXBLN     INDICATE ZERO BUFFER LENGTH        @
         MVI   PXUBTYP,0           CLEAR USER DATA IN XPCCB           @
         MVI   PXUACT1,PXUATRMR    INDICATE RETURN QUEUED MESSAGES    @
         BAL   RD,SENDR            GO TO SENDR ROUTINE                @
         CLI   PXPRETCD,PXPRCOK    WAS POWER RETURN CODE ZERO?        @
         BNE   REQFAIL             NO, GO TO HANDLE REQUEST FAILURE   @
         BAL   RE,DATDSPLY         YES, GO TO DISPLAY RETURNED MSG'S  @
         SPACE 1                                                     @
```

Page 17 of 26

```
DISPLAY  DS    0H                                                     @
         DPLAY SUCCM3,72           DISPLAY MESSAGE                    @
         DPLAY SUCCM4,72           DISPLAY MESSAGE                    @
         DPLAY SUCCM6,72           DISPLAY MESSAGE                    @
         SPACE 1                                                      @
         B     DISCT               DISCONN AND TERMIN XPCC LINK, EOJ  @
         EJECT                                                        @
**********************************************************************@
**                  >>   DATDSPLY ROUTINE       <<                   *@
**  THIS ROUTINE DISPLAYS MESSAGES AND DATA RETURNED BY VSE/POWER.   *@
**********************************************************************@
         SPACE 1                                                      @
*        REGISTER USAGE FOR DATDSPLY ROUTINE                          @
         SPACE 2                                                      @
*        RA - BUFPTR  - POINTER FOR THE REPLY BUFFER                  @
*        RC - BUFLN   - REG TO CALCULATE THE LENGTH OF THE DATA STILL @
*                       TO BE DISPLAYED                               @
*        R0, R1, R2, R3 - WORK REGISTER                               @
*                                                                     @
*        CALLED FROM: PUT REQUEST TO RDR QUEUE                        @
*                     CTL REQUEST                                     @
*                     GET REQUEST                                     @
*                     PUT REQUEST TO LST QUEUE                        @
*                                                                     @
*        EXIT TO CALLER IF ALL AVAILABLE MESSAGES/DATA ARE DISPLAYED  @
         SPACE 2                                                      @
DATDSPLY DS    0H                                                     @
         SR    R0,R0               SET R0 TO ZERO                     @
         CLM   R0,M7,IJBXSLN       NO MORE DATA TO DISPLAY?           @
         BER   RE                  RETURN TO CALLER                   @
         LA    BUFPTR,REPLBUF      POINT TO REPLY BUFFER              @
         SR    BUFLN,BUFLN         CLEAR REGISTER                     @
         ICM   BUFLN,M7,IJBXSLN    GET LENGTH OF DATA TO BE DISPLAYED @
         SPACE 1                                                      @
*   PWRSASEX DISPLAYS, RECORD AFTER RECORD, THE DATA OR MESSAGES RE-  @
*   TURNED BY VSE/POWER.  THE RECORD PREFIX OF EACH DATA RECORD IS    @
*   ANALYZED BUT NOT DISPLAYED.                                       @
*   IF DATDSPLY IS CALLED BY THE GET FUNCTION, THE LOGICAL RECORD     @
*   NUMBER OF THE 12TH DATA CARD OF JOB 'EXAMPLE' IS SAVED.  THE PRO- @
*   GRAM USES THIS NUMBER LATER AS A RESTART POINT.                   @
         SPACE 1                                                      @
```

```
                                                              Page 18 of 26

DSPL0    DS    0H                                                     @
         USING RECPRFIX,BUFPTR     GET DSECT OF RECORD LAYOUT         @
         CLI   GETFCT,C'G'          WAS DATDSPLY CALLED BY GET?       @
         BNE   DSPL1                NO, GO TO DSPL1                   @
         CLC   RECPRFXL(4,BUFPTR),=C'* 12'  IS THE CURRENT CARD NO.12 @
         BNE   DSPL1                        NO, GO TO DSPL1           @
         MVC   PWRRECNO,RECLOGNO   SAVE LOGICAL RECORD NUMBER         @
         SPACE 1                                                      @
DSPL1    DS    0H                                                     @
         LH    R2,RECLNGTH         GET LENGTH OF FIRST/NEXT DATA REC. @
         LA    BUFPTR,RECPRFXL(,BUFPTR)     SKIP RECORD PREFIX        @
         SPACE 1                                                      @
         DPLAY (BUFPTR),(R2)       DISPLAY CURRENT DATA RECORD        @
         SPACE 1                                                      @
         LA    R1,RECPRFXL(,R2)    CALC. LENGTH OF RECORD INCL. PREFIX@
         SR    BUFLN,R1            CALC.LENGTH OF DATA STILL IN BUFFER@
         LA    BUFPTR,0(R2,BUFPTR)     POINT TO NEXT RECORD           @
         LTR   BUFLN,BUFLN         ALL DATA IN BUFFER DISPLAYED?      @
         BNZ   DSPL0               NO, GO TO DISPLAY NEXT DATA REC.   @
         SPACE 1                                                      @
         CLI   PXPFBKCD,PXP00EOD   END OF DATA?                       @
         BER   RE                  YES, RETURN TO CALLER              @
         SPACE 1                                                      @
*   IF THIS ROUTINE IS CALLED BY THE GET FUNCTION, 'SEND (MORE) DATA' @
*   HAS TO BE INDICATED IN THE ACTION BYTE. IN ALL OTHER CASES        @
*   'RETURN (MORE) MESSAGES' MUST BE SET.                             @
         SPACE 1                                                      @
DSPL2    DS    0H                                                     @
         XC    IJBXBLN,IJBXBLN     INDICATE ZERO BUFFER LENGTH        @
         MVI   PXUBTYP,0            CLEAR BUFFER TYPE BYTE            @
         MVI   PXUACT1,PXUATRMR    INDICATE A 'RETURN MESSAGE' REQUEST@
         CLI   GETFCT,C'G'          WAS DATDSPLY CALLED BY GET?       @
         BNE   DSPL3                NO, KEEP RETURN MESSAGE INDICATION@
         MVI   PXUACT1,PXUATSDR    INDICATE A 'SEND DATA' REQUEST     @
DSPL3    DS    0H                                                     @
         MVC   FAILLABL,=C'DSPL2'  INSERT CODE LABEL FOR DIAGNOSTIC   @
         BAL   RD,SENDR            GO TO SENDR ROUTINE                @
         CLI   PXPRETCD,PXPRCOK    WAS POWER RETURN CODE ZERO?        @
         BNE   REQFAIL             NO, GO TO HANDLE REQUEST FAILURE   @
         B     DATDSPLY            YES, START DISPLAYING AGAIN        @
         EJECT                                                        @
```

```
                                                             Page 19 of 26

***********************************************************************@
**      >> ROUTINE TO HANDLE REQUEST FAILURES         <<              *@
**  THE ROUTINE IS CALLED IF POWER RC/FBKC WAS NOT ZERO               *@
***********************************************************************@
         SPACE 1                                                       @
REQFAIL  DS    0H                                                      @
         MVC   FAILFUNC,=C'SENDR   ' INSERT FAILING FUNCTION INTO MSG @
         BAL   RE,MSGRCFB          PREPARE RC/FBK CODE DISPLAY         @
         BAL   RE,MSGDSPLY         DISPLAY MESSAGE ON CONSOLE          @
         B     FINDUMP             GO TO TERMINATION WITH DUMP         @
         EJECT                                                         @
************************************************************************
**    >> DISCONNECT THE XPCC COMMUNICATION LINK TO VSE/POWER <<       **
** IF THE MACRO FAILS, THE PROGRAM DISPLAYS A DIAGNOSTIC MESSAGE AND **
** TERMINATES WITH A DUMP.                                            **
************************************************************************
         SPACE 1
DISCT    DS    0H
         XPCC  XPCCB=(R4),FUNC=DISCONN   DISCONNECT LINK TO VSE/POWER
         SPACE 1
         LTR   RF,RF               WAS DISCONNECT SUCCESSFUL, RF='00' ?
         BZ    TERMN               ..YES CONTINUE WITH XPCC TERMINATION
         SPACE 1
         MVC   FAILFUNC,=C'DISCONN '   INSERT FAILING FUNCTION
         BAL   RE,MSGRETC          INSERT XPCC RETURN CODE INTO MSG
         MVC   FAILLABL,=C'DISCT'  INSERT CODE LABEL FOR DIAGNOSTIC
         BAL   RE,MSGDSPLY         DISPLAY MESSAGE ON CONSOLE
         B     FINDUMP             GO TO TERMINATION WITH DUMP
         EJECT
************************************************************************
**    >> TERMINATE INTERACTION WITH THE VSE/AF XPCC SUPPORT <<        **
** IF THE MACRO FAILS, THE PROGRAM DISPLAYS A DIAGNOSTIC MESSAGE AND **
** TERMINATES WITH A DUMP.                                            **
************************************************************************
         SPACE 1
TERMN    DS    0H
         XPCC XPCCB=(R4),FUNC=TERMIN   TERMINATE CROSS PART. INTERFACE
         LTR   RF,RF               DID WE GET A ZERO RET-CODE ?
         BZ    FINEND              ..YES, GO TO NORMAL EOJ MACRO
         SPACE 1
         MVC   FAILFUNC,=C'TERMIN  '  INSERT FAILING FUNCTION INTO MSG
         BAL   RE,MSGRETC          INSERT XPCC RETURN CODE INTO MSG
         MVC   FAILLABL,=C'TERMN'  INSERT CODE LABEL FOR DIAGNOSTIC
         BAL   RE,MSGDSPLY         DISPLAY MESSAGE ON CONSOLE
         B     FINDUMP             GO TO TERMINATION WITH DUMP
         EJECT
```

Page 20 of 26

```
************************************************************************
**                  >> TERMINATE PWRSASEX  <<                         **
************************************************************************
         SPACE 1
FINDUMP  DS    0H                    TERMINATION FORCED DUE TO ERROR
*        DUMP                        A PARTITION DUMP CAN BE FORCED IF
*                                    NECESSARY FOR DEBUG PURPOSES
         SPACE 1
FINEND   DS    0H                    NORMAL TERMINATION
         EOJ                         NORMAL END OF PWRSASEX PROGRAM
         EJECT
************************************************************************
**     >>       MESSAGE BUILD ROUTINE FOR FAILMSG            <<       **
** BRANCHED TO FROM ANY CALLER TO FILL SELECTED FIELDS OF THE DIAG-   **
** NOSTIC MESSAGE.  RETURNS TO CALLER VIA REGISTER 14 (RE).           **
************************************************************************
         SPACE 1
MSGRETC  DS    0H
         UNPK  HELP,IJBXRETC(2)          UNPACK HEX XPCC RETURN CODE
         TR    HELP(2),TRTAB             CONVERT TO PRINTABLE HEX-VALUE
         MVC   FAILRETC,HELP             INSERT PRINTABLE XPCC RET. CODE
         BR    RE                        RETURN TO CALLER
         SPACE 1
MSGREAS  DS    0H
         UNPK  HELP,IJBXREAS(2)          UNPACK HEX XPCC REASON CODE
         TR    HELP(2),TRTAB             CONVERT TO PRINTABLE HEX-VALUE
         MVC   FAILREAS,HELP             INSERT PRINTABLE XPCC REAS. CODE
         BR    RE                        RETURN TO CALLER
         SPACE 1
MSGRCFB  DS    0H
         UNPK  HELP,PXPRETCD(2)          UNPACK HEX POWER RETURN CODE
         TR    HELP(2),TRTAB             CONVERT TO PRINTABLE HEX-VALUE
         MVC   FAILPWRC,HELP             INSERT PRINTABLE POWER RET. CODE
         SPACE 1
         UNPK  HELP,PXPFBKCD(2)          UNPACK HEX POWER FEEDBACK CODE
         TR    HELP(2),TRTAB             CONVERT TO PRINTABLE HEX-VALUE
         MVC   FAILPWFB,HELP             INSERT POWER FEEDBACK CODE
         BR    RE                        RETURN TO CALLER
         SPACE 1
MSGDSPLY DS    0H
         DPLAY FAILMSG,72                   DISPLAY FAILURE MESSAGE
         SPACE 1
         BR    RE
         EJECT
```

Page 21 of 26

```
***********************************************************************
**            >> CENTRAL XPCC SENDR ROUTINE  <<                      **
**  BEFORE THIS ROUTINE IS CALLED, THE PROGRAM INSERTS THE CALLING   **
**  POINT IN THE DIAGNOSTIC MESSAGE THAT IS ISSUED SHOULD THE SENDR  **
**  MACRO FAIL.  THIS ROUTINE:                                       **
**   - ISSUES THE XPCC MACRO WITH FUNC=SENDR AND WAITS FOR THE       **
**     SECB TO BE POSTED.  IT CHECKS REGISTER 15 (RF) AND THE VSE    **
**   - CHECKS REGISTER 15 (RF) AND THE VSE RETURN- AND REASON CODES  **
**     IN FIELDS IJBXRETC AND IJBXREAS, RESPECTIVELY.                **
**   - CHECKS THE VSE/POWER RETURN CODE IN FIELD PXPRETCD IF         **
**     VSE/POWER DISCONNECTS THE COMMUNICATION PATH WITH A PURGE.    **
**  THE ROUTINE RETURNS TO THE CALLER IF THE XPCC MACRO CALL COM-    **
**  PLETED SUCCESSFULLY OR, IN CASE OF A FAILURE, THE VSE/POWER RE-  **
**  TURN CODE IS NOT TOO SEVERE.  RETURN IS PROVIDED VIA REGISTER    **
**  13 (RD).                                                         **
***********************************************************************
         SPACE 1                                                      @
*        REGISTER USAGE FOR SENDR ROUTINE                             @
         SPACE 2                                                      @
*        R3 - WORK REGISTER (FOR WAIT)                                @
*        RD - REGISTER USED TO RETURN TO CALLER                       @
*                                                                     @
*        CALLED FROM: PUT REQUEST TO RDR QUEUE                        @
*                     CTL REQUEST                                     @
*                     GET REQUEST                                     @
*                     PUT REQUEST TO LST QUEUE                        @
*                     DATDSPLY ROUTINE                                @
*                                                                     @
*        EXIT TO CALLER (SEE COMMENT ABOVE)                           @
*                OR TO DISCT OR FINDUMP IN CASE OF A FAILURE          @
*                                                                     @
         SPACE 2                                                      @
         SPACE 1
SENDR    DS    0H
         XPCC  XPCCB=(R4),FUNC=SENDR      SEND BUFFER TO VSE/POWER
         LTR   RF,RF                     DID WE GET A ZERO RETURN CODE ?
         BZ    WAITSECB                  ..YES, THEN WAIT FOR REPLY OF POWER
         SPACE 2
```

```
*   IF THE SENDR MACRO COMPLETES WITH RF=X'08', THEN THE ROUTINE:
*   1.  FILLS THE DIAGNOSTIC MESSAGE ACCORDING TO THE VSE RETURN CODE.
*   2.  DISPLAYS THE MESSAGE.
*   3.  TERMINATES WITH OR WITHOUT A DUMP.
*   THERE IS NO RETURN TO THE CALLER OF SENDR.
         SPACE 1
TESTRETC DS    0H
         CLI   IJBXRETC,IJBXNOC3   DID POWER ABNORMALLY TERMINATE ?
         BE    ABNPOW              ..YES, THEN GO TO STOP PWRSASEX
         MVC   FAILFUNC,=C'SENDR  '    INSERT 'SENDR ' INTO MSG TEXT
         BAL   RE,MSGRETC          PUT XPCC RETURN CODE INTO MSG
         CLI   IJBXRETC,IJBXNOC2   DID POWER GIVE A DISCONNECT PURGE ?
         BE    TERMCONN            ..YES,THEN GO TO SHOW WHY, TERMINATE
         BAL   RE,MSGDSPLY         DISPLAY DIAGNOSTIC MESSAGE ON CONS.
         B     FINDUMP             TERMINATE PWRSASEX WITH PART.DUMP
         SPACE 1
ABNPOW   DS    0H
         DPLAY FAILM2,72
         SPACE 1
         B     DISCT               DISCONN AND TERMIN XPCC LINK, EOJ
         SPACE 2
*   THE ROUTINE WAITS FOR THE SEND ECB TO BE POSTED.  IT RETURNS TO THE
*   CALLER IF THE SYSTEM PASSED A REASON CODE OF ZERO, THAT IS, THE
*   XPCC CONNECTION IS ERROR FREE.
*   FOR A NON-ZERO REASON CODE, THE ROUTINE DISPLAYS A DIAGNOSTIC
*   MESSAGE AND TERMINATES WITH OR WITHOUT A DUMP.
         SPACE 1
WAITSECB DS    0H
         LA    R3,IJBXSECB         LOAD ADDRESS OF SEND COMPLETION ECB
         WAIT  (R3)                WAIT FOR COMPLETION  OF SENDR
         CLI   IJBXREAS,REASOK     DID ANY CONNECTION ERROR OCCUR ?
         BER   RD                  .. NO, THEN RETURN TO CALLER
         SPACE 1
BADREAS  DS    0H
         TM    IJBXREAS,IJBXABDC   DID POWER TERMINATE ABNORMALLY ?
         BO    ABNPOW              .. YES, GIVE MESSAGE AND GO TO EOJ
         MVC   FAILFUNC,=C'SENDR  '    INSERT 'SENDR ' INTO MSG TEXT
         BAL   RE,MSGREAS          FILL XPCC REASON CODE INTO MSG
TERMCONN DS    0H
         BAL   RE,MSGRCFB          PUT POWER RETURN/FEEDBACK TO MSG
         BAL   RE,MSGDSPLY         DISPLAY DIAGNOSTIC MESSAGE
         CLI   PXPRETCD,PXPRCPVL   POWER RC = PROTOCOL VIOLATION?
         BE    FINDUMP             .. YES, USER ERROR
         B     DISCT               ..YES, SYSTEM ERROR
         EJECT
```

Page 23 of 26

```
**********************************************************************
**                D E F I N I T I O N S                             **
**********************************************************************
         SPACE 2
TRTAB    EQU   *-240               ENTRY POINT FOR TRANSLATE TABLE
         DC    X'F0F1F2F3F4F5F6F7F8F9C1C2C3C4C5C6' TRANSLATE TABLE
         SPACE 1
EIGHTDC  DC    X'08'               BYTE TO TEST RETURN CODE
HELP     DC    CL3' '              FIELD FOR UNPACK RET CODE
         SPACE 1
WAITLIST DC    A(INTECB)           INTECB = 1ST ELEMENT OF WAITLIST
LISTCECB DC    A(0)                IJBXCECB = 2ND ELEM. OF WAITLIST
LISTEND  DC    X'FF'               INDICATE END OF WAITLIST
         SPACE 1
INTECB   DS    F                   ECB USED TO WAIT FOR TIMER INTERVALS
         SPACE 1
EIGHT    EQU   X'08'               RETURN CODE X'08'
POSTBIT  EQU   X'80'               MASK FOR A POSTED ECB
REASOK   EQU   X'00'               ZERO VSE/AF REASON CODE
         EJECT
**********************************************************************@
*            DEFINITIONS FOR PUT,CTL AND GET REQUEST                 *@
**********************************************************************@
         SPACE 1                                                      @
M1       EQU   1                   MASK BIT SETTING                   @
M7       EQU   7                   MASK BIT SETTING                   @
ZERO     EQU   0                                                      @
ONE      EQU   1                                                      @
         SPACE 1                                                      @
NOOFRECS EQU   7                   NUMBER OF RECORDS IN A SEGMENT     @
DISP     EQU   C'L'                DISPOSITION OF OUTPUT TO BE SENT   @
PRIOR    EQU   C'9'                PRIORITY OF OUTPUT TO BE SENT      @
SEPPAGE  EQU   4                   NUMBER OR SEPARATOR PAGES/CARDS    @
         SPACE 1                                                      @
RECORDCT EQU   2                   USE R2 AS REC COUNTER IN LOOP      @
BUFPTR   EQU   10                  USE RA AS BUFPOINTER               @
DATAPTR  EQU   11                  USE RB AS DATA POINTER             @
BUFLN    EQU   12                  USE RC TO CALC REMAINING BUFLEN    @
         SPACE 2                                                      @
JOBNAME  DS    CL8                 FIELD TO SAVE JOBNAME RET'D BY POW.@
JOBNUM   DS    CL2                 FIELD TO SAVE JOBNUMB.RET'D BY POW.@
PWRRECNO DS    F                   FIELD TO SAVE POW. LOGICAL REC NO. @
GETFCT   DC    C' '                FIELD TO IDENTIFY GET AS CALLER OF @
*                                                          DATDSPLY   @
         SPACE 1                                                      @
FORMS    DC    CL8'AABB'           FORMS OF OUTPUT TO BE SENT         @
JOBNLAB  DC    CL8'EXAMPSEG'       NAME OF OUTPUT TO BE SPOOLED       @
         EJECT                                                        @
```

Page 24 of 26

```
***********************************************************************
*            MESSAGE AREA FOR FAILING MACRO CALLS                     *
***********************************************************************
         SPACE 1
FAILMSG  DS    0CL72
F1       DC    C'FUNC='
FAILFUNC DC    CL8' '                REQUESTED FUNCTION
F2       DC    C' FAILED AT: '
FAILLABL DC    CL5' '                CODE LABEL OF FAILING FUNCTION
F3       DC    C'  VSE-RETC/REAS='
FAILRETC DC    CL2'00'               RETURN CODE RECEIVED IN IJBXRETC
F4       DC    C'/'
FAILREAS DC    CL2'00'               REASON CODE RECEIVED IN IJBXREAS
F5       DC    C' PWR-RC/FDBK='
FAILPWRC DC    CL2'00'               VSE/POWER RETURN CODE IN IJBXRUSR
F6       DC    C'/'
FAILPWFB DC    CL2'00'               VSE/POWER FEEDBACK CODE IN IJBXRUSR
F7       DC    CL3' '
         SPACE 1
FAILM1   DC    CL72'VSE/POWER ALREADY IN TERMINATION, NO MORE CONNECTIO*
               N REQUEST ACCEPTED'
FAILM2   DC    CL72'VSE/POWER ABNORMAL TERMINATION, CONNECTION DISRUPTE*
               D'
FAILM3   DC    CL72'CONNECTION COULD NOT BE COMPLETED WITHIN 2 MINUTES'
FAILM4   DC    CL72'LIST QUEUE ENTRY COULD NOT BE FOUND, PWRSASEX WILL *
               STOP'
         SPACE 1
SUCCM1   DC    CL72'>>> XPCC CONNECTION TO VSE/POWER SUCCESSFULLY BUILT*
                <<<'
SUCCM2   DC    CL72'>>> NOW PWRSASEX WILL RESTART ON RECORD NO.12 <<<'
SUCCM3   DC    CL72'>>> THE VSE/POWER LIST QUEUE MUST NOW CONTAIN 3 SEG*
               MENTS.. <<<'
SUCCM4   DC    CL72'>>> .... NAMED EXAMPSEG AND A SINGLE ENTRY NAMED EX*
               AMPLE ] <<<'
SUCCM6   DC    CL72'>>> *** SUCCESSFUL TERMINATION OF PWRSASEX *** <<<'
SUCCM7   DC    CL72'>>> NOW FOLLOWS THE DISPLAY OF THE LIST ENTRY: EXAM*
               PLE <<<'
SUCCM8   DC    CL72'>>> NEXT PWRSASEX WILL SUBMIT DATA TO THE LIST QUEU*
               E <<<'
         EJECT
```

Page 25 of 26

```
***********************************************************************
*         JOB 'EXAMPLE' TO BE PASSED TO VSE/POWER                     *
***********************************************************************
         SPACE 1
JECL1    DC    C'* $$ JOB JNM=EXAMPLE,DISP=D,CLASS=A                '
JECL2    DC    C'* $$ LST DISP=K,CLASS=S                            '
JCL1     DC    C'// JOB EXAMPLE                                     '
DATACARD DC    C'* 01---------------------*---------------------01 *'
         DC    C'* 02-------------------*   *-------------------02 *'
         DC    C'* 03-----------------*       *-----------------03 *'
         DC    C'* 04---------------*           *---------------04 *'
         DC    C'* 05-------------*               *-------------05 *'
         DC    C'* 06-----------*                   *-----------06 *'
         DC    C'* 07---------*                       *---------07 *'
         DC    C'* 08-------*                           *-------08 *'
         DC    C'* 09-----*                               *-----09 *'
         DC    C'* 10---*                                   *---10 *'
         DC    C'* 11-*                                       *-11 *'
         DC    C'* 12---*                                   *---12 *'
         DC    C'* 13-----*                               *-----13 *'
         DC    C'* 14-------*                           *-------14 *'
         DC    C'* 15---------*                       *---------15 *'
         DC    C'* 16-----------*                   *-----------16 *'
         DC    C'* 17-------------*               *-------------17 *'
         DC    C'* 18---------------*           *---------------18 *'
         DC    C'* 19-----------------*       *-----------------19 *'
         DC    C'* 20-------------------*   *-------------------20 *'
         DC    C'* 21---------------------*---------------------21 *'
JCL2     DC    C'/&&                                                '
JECL3    DC    C'* $$ EOJ                                           '
ENDIND   DC    C'/+'
         EJECT
***********************************************************************
*        CROSS PARTITION CONTROL BLOCK                                *
***********************************************************************
         SPACE 1
OWNXPCCB XPCCB APPL=PWRSASEX,TOAPPL=SYSPWR,                            *
               BUFFER=(SENDBUF,400),REPAREA=(REPLBUF,500)
         SPACE 2
***********************************************************************
*        STORAGE RESERVATION FOR  XPCC SEND AND REPLY BUFFER          *
***********************************************************************
         SPACE 1
SENDBUF  DS    CL400              BUFFER USED FOR XPCC SENDR TO POWER
LASTPREC DC    A(SENDBUF+L'SENDBUF-RECPRFXL-L'DATACARD) LAST POSSIBLE
*                                 RECORD THAT FITS INTO SEND BUFFER
REPLBUF  DS    CL500              BUFFER FOR RECEIPT OF DATA FROM POWER
         EJECT
```

```
                                                          Page 26 of 26

************************************************************************
         LTORG
************************************************************************
         EJECT
***********************************************************************@
**      >>           GENERATE   S P L                   <<             *@
**       THIS SPL IS LATER ON UPDATED IN ORDER TO INDICATE A          *@
**       GET, PUT, OR CTL REQUEST WITH THE DESIRED PARAMETERS         *@
***********************************************************************@
         SPACE 1                                                       @
OWNSPL   PWRSPL TYPE=GEN,USERID=SASUSER1,PRFX=OWN                      @
         EJECT                                                         @
************************************************************************
*        DUMMY SECTION OF  VSE/POWER SPOOL PARAMETER LIST (SPL)        *
************************************************************************
         SPACE 1
OWNSPLDS PWRSPL TYPE=MAP
         EJECT
************************************************************************
*        DUMMY SECTION OF  CROSS PARTITION CONTROL BLOCK  (XPCCB)      *
************************************************************************
         SPACE 1
         MAPXPCCB
         EJECT
************************************************************************
*        GENERAL EQUATES                                               *
************************************************************************
         SPACE 1
R0       EQU   0                   WORK REGISTER
R1       EQU   1                   WORK REGISTER + USED BY PWRSPL MACRO
R2       EQU   2                   WORK REGISTER
R3       EQU   3                   WORK REGISTER
R4       EQU   4                   ADDR REG FOR XPCCB DSECT
R5       EQU   5                   ADDR REG FOR USER DATA TO BE SENT
R6       EQU   6                   ADDR REG FOR RECEIVED USER DATA
R7       EQU   7                   ADDR REG FOR SPL DSECT
R8       EQU   8                   FIRST BASE REGISTER OF PWRSASEX
R9       EQU   9                   SECOND BASE REGISTER OF PWRSASEX
RA       EQU   10                  WORK REGISTER
RB       EQU   11                  WORK REGISTER
RC       EQU   12                  WORK REGISTER
RD       EQU   13                  BRANCH AND LINK REGISTER FOR SENDR
RE       EQU   14                  BRANCH AND LINK REG. FOR DATDSPLY
RF       EQU   15                  MACRO CALL RETURN CODE REGISTER
         SPACE 1
```

CHAPTER 3. EXTERNAL DEVICE SUPPORT

This support is a dedicated application of the spool-access support described in Chapter 2; give that chapter a thorough reading before you study this chapter.

The support shifts the control for writing spooled output to a device from VSE/POWER to a subsystem (or application program), occasionally also referred to as device-driving system (DDS). This subsystem may run in a partition under or outside the control of VSE/POWER. The support allows you, for example, to process output spooled to the LST or PUN queue on a device which is not supported by VSE/POWER.

Using the support requires you to implement extensive coding of your own in your program. This coding must be done in assembler language.

This chapter briefly discusses the operational concepts of the support and describes how to use it. The macros you need to implement the support in your program are documented in Chapter 2.

CONCEPTS

Programming Prerequisites: Figure 40 on page 150 shows how a device controlling subsystem communicates with VSE/POWER. Before a subsystem-controlled device can be started for output of spooled data, this subsystem must:

1. Identify itself to the system.

2. Issue one or more connect-any requests, one per device that is to be used for the processing of spooled output. A connect-any request ensures that VSE/POWER can establish a communication path when a PSTART command for the device is issued.

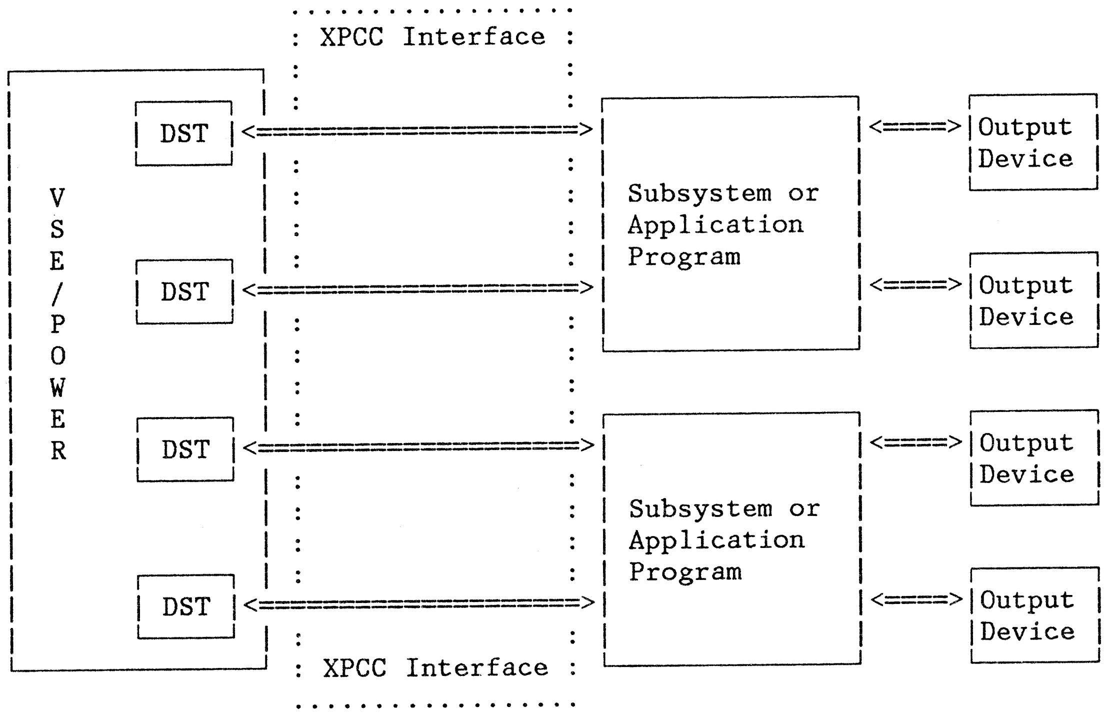
User Responsibilities: The subsystem must provide for all of the services normally available for a device under VSE/POWER control. This includes services such as device recovery, measurement techniques for performance and accounting, and protection of spooled data after VSE/POWER has passed this data to the subsystem.

Operational Overview: Following is an overview of the operational steps involved in writing spooled output to a device under subsystem control. This overview assumes that the device-owning subsystem is up and running. It further assumes that the output device to be used is ready.

1. VSE/POWER processes a PSTART command for the device – for example:

```
PSTART DEV,PLOT1,GRAPHAPP,G,...
```

The command causes VSE/POWER to activate a device service task (DST) which, in turn, establishes a communication path to the subsystem named GRAPHAPP.



Legend: =====> Data flow (includes control data)
DST = Device service task

Figure 40. External Device Support Overview

2. When the communication path is established, the device owning subsystem passes to VSE/POWER a request for a device order.

3. In response to the request, VSE/POWER passes to the subsystem a "start device" order. This order includes all of the control values that were specified in the above PSTART command.

4. The subsystem, after having confirmed the order by passing an order-response record, would normally issue a GET-GENERIC request. An example of such a request is given below:

```
PWRSPL TYPE=UPD,CLASS=G,MODE=GENERIC,QUEUE=LST,REQ=GET,SPL=MYSPL...
```

Passing this updated SPL to VSE/POWER via XPCC FUNC=SENDR causes VSE/POWER to return to your program any output queued with the specified class (G in the above example) of the LST queue for the device PLOT1.

The retrieval of a complete queue entry requires the subsystem to issue a series of Get spool data requests with XPCC FUNC=SENDR. Per request, VSE/POWER passes a unit of transfer, one or more records of data, to your program's reply buffer.

Note that before your program can set up an XPCC FUNC=SENDR request, it must always clear the XPCCB User Data IJBXSUSR.

5. The subsystem writes each unit of transfer to the output device selected by the PSTART command.

6. When the processing of a queue entry is complete, the subsystem issues a close request followed by another GET-GENERIC request to open the retrieval of the next eligible queue entry.

The above sequence of operational steps continues as long as there is work to do. This sequence, although not all inclusive, shows that your program must synchronize its operation with VSE/POWER primarily by:

1. Picking up and analyzing any device order that VSE/POWER may pass.

2. Responding to a device order by passing to VSE/POWER the corresponding order-response record. This response record must indicate how your program is going to handle the device order.

Shared Spooling Considerations: For operation with external device support in a shared spooling environment, the following restrictions exist:

- Only one system operator can control your program's output devices: the operator of the system on which your program is running.

- Messages passed to VSE/POWER for routing to a user of one of the other sharing systems cannot be forwarded to this user by VSE/POWER.

The remaining sections of this chapter discuss the sequences of the coding required to ensure proper handling of spooled output. These sequences are discussed as part of the applicable communication and device-control functions.

IBM recommends that you obtain a listing of the DSECTS that are generated by the assembly of the PWRSPL TYPE=MAP macro and that you have this listing readily available at your finger tips. This may be helpful for the study of the chapter.

SET UP A COMMUNICATION PATH

Your program must initiate the setup of required communication paths. To do this, provide code in your program to:

1. Identify your program to the system.

   You do this by way of an XPCC macro specifying FUNC=IDENT.

2. Initiate setting up a communication path, one per device.

   You do this by way of an XPCC FUNC=CONNECT with TOAPPL=ANY specified in the related XPCCB macro.

   In your program, you can issue as many XPCC FUNC=CONNECT requests as you have devices to control for the processing of spooled output. A connect request must be complete before you can issue the next one.

For more information about establishing a communication path, see the section "Set Up a Communication Path" on page 33. The section "Set Up Two or More Communication Paths" on page 97 describes how to establish two or more communication paths.

START A DEVICE

Starting a device is triggered by a PSTART command issued by one of the following:

The central operator.
An authorized subsystem administrator.
Via PNET.

In processing the command, VSE/POWER tries to set up a communication path within two minutes. If VSE/POWER cannot set up the path within this time, then the originator of the PSTART command receives a message.

VSE/POWER expects this originator to respond to the message. This response may be an instruction to wait until the subsystem is prepared for this setup or to stop the device by a PSTOP DEV command.

Your program must include code which does the following (assuming that you have properly initiated the setup of a communication path):

1. Waits for the communication path to be set up.

   You do this by checking whether the system has posted the connect ECB (field IJBXCECB of the applicable XPCCB).

2. Passes to VSE/POWER a request for a device order.

   You do this by issuing an XPCC FUNC=SENDR request which:

- Passes a null buffer

- Has XPCCB bytes set as follows:

  PXUACT1 to PXUATROR
  PXUBTYP to zero

  and by checking for successful completion or, if necessary, by analyzing return information that the system may have set in the fields IJBXRETC and later in IJBXREAS of the XPCCB.

3. Analyzes the start device order which VSE/POWER passes in the reply buffer for the communication path.

4. Passes to VSE/POWER the corresponding order-response record.

   To do this, issue an XPCC FUNC=SENDR request with this record set up in the communication path's send buffer. Before you issue this request, clear the XPCCB User Data IJBXSUSR.

For a more detailed discussion of the start-device sequence, see the related sections that follow.

Process a Start-Device Order

The Device Can be Started: Refer to Figure 41 on page 155, the coding sequence for starting a device under subsystem control. For the layout and contents of order-control and response records, see the section "Process Order-Control Records and Signals" on page 176.

The Device Cannot Be Started: Your program may not be prepared to process output on the device as requested. You must indicate this and give a reason by setting a return and feedback code in your order-response record for one of the following, for example:

Device unknown
Device in use (busy)
Device out of service

A return code other than X'00' causes VSE/POWER to break the connection. For details about these codes, see the section "Start-Device Order" on page 184.

Based on your program's control data, VSE/POWER builds a message and routes it to the command originator.

Start a Device with Setting Logical Destinations

If your program does not use a set-logical-destinations order, VSE/POWER takes the specified device name (PLOT1 for example) as the only valid destination name for the device.

If you use a set-logical-destinations order, your program can define to VSE/POWER up to eight logical destination names for one device. Assume that a device has been started in your program with a device name of PLOT1. You could then request VSE/POWER by a set-logical-destinations order to route, via the path for PLOT1, output to the following destinations, for example:

D121OUT
D122OUT
D123OUT
and so on

If any of these logical destinations is specified as destination user of an output, then VSE/POWER routes this output to the external device named PLOT1.

However, if the device name used in the PSTART command is to be used as userid for routing output further on, that name must be included in the list of logical destinations.

Note: The logical destination name LOCAL returns queue entries either destined for local processing or destined for the userid LOCAL.

```
     Coding in Your Program                   Comments
 ________________________________     ________________________________

   ...
   XPCC FUNC=CONNECT                  Connect with TOAPPL=ANY
     Check the return codes in
      register 15 and in the
      XPCCB (byte IJBXRETC).
   WAIT IJBXCECB                      The communication path exists when
        |                             the ECB is posted.
        V                             Field IJBXTOAP of XPCCB contains
 Request a device order to be passed  'SYSPWRD'.
   XPCC FUNC=SENDR
     Check the return codes
      as shown above.
   WAIT IJBXSECB                      A device order is in your program's
     Check the VSE reason codes       reply buffer when the ECB is posted.
      in XPCCB byte IJBXREAS.
     Check the VSE/POWER return
      and feedback codes in XPCCB
      bytes PXPRETCD and PXPFBKCD,
      respectively.
     Analyze the device order         Normally, your program finds a start
        |                             device order after successful setup
        V                             of a communication path.
 Respond to the order
   XPCC FUNC=SENDR
     Check the return codes
      as shown above.
   WAIT IJBXSECB                      VSE/POWER has finished processing
     Check the VSE reason codes       your order-response record and returned
      in XPCCB byte IJBXREAS.         a null buffer when the ECB is posted.
     Check the VSE/POWER return          If these codes indicate success-
      and feedback codes as           ful processing of the order-response
      shown above.                    record, then VSE/POWER is ready to
        |                             process GET-service requests.
        V
```

Figure 41. Coding Sequence for Starting an External Device

The coding sequence for a device start with setting logical destinations is the same as for a normal device start (see Figure 41). In addition, however, your program must pass to VSE/POWER a set-logical-destinations order. You do this after VSE/POWER has successfully processed your order-response record for the start-device order. VSE/POWER responds to your order by passing an order-response record to your program's reply buffer.

For the layout and contents of control records, see the section "Process Order-Control Records and Signals" on page 176.

PROCESS SPOOLED OUTPUT

When your program is ready to process an output queue entry, it should issue a generic GET-service request. To do this, open the GET-service request by passing to VSE/POWER an SPL for which you defined, for example, the following:

```
                                                Column 72 ───┐
                                                             V
    PWRSPL  TYPE=UPD,SPL=(4),CLASS=G,MODE=GENERIC,           C
            QUEUE=LST,REQ=GET
```

VSE/POWER then retrieves from the accessed queue (LST in the example) the first queue entry that it finds to have:

- Class G assigned.
- A disposition of D or K.
- A user identifier matching one of the logical destinations of the device.

In response to your open-service request, VSE/POWER passes to your program's reply buffer an SPL which describes the queue entry's characteristics. Your program must analyze this SPL and decide whether VSE/POWER is to proceed with data retrieval or whether any other action is to be initiated.

The subsystem, your program, has to handle certain situations which VSE/POWER handles when processing the output of spooled data on a local device. The handling of these situations is normally triggered by a device order passed to your program by VSE/POWER. Of course, the handling of a device failure, should one occur, cannot be triggered by VSE/POWER. Some of these situations are discussed in sections as indicated below; they should give you a feel for the involved programming effort:

- No selectable entry in the accessed queue – See "Handle a No-Selectable-Entry Situation" on page 157.
- A device setup is required to process the output – See "Handle a Device-Setup Situation" on page 157.
- Output processing is to be canceled – See "Cancel Output Processing" on page 163.
- VSE/POWER-queued device orders or signals are to be requested – See "Request an Order or a Signal" on page 164.

Note: No password checking is done for a queue entry that is to be processed by a subsystem for output under subsystem control.

Handle a No-Selectable-Entry Situation

If there is no selectable queue entry, VSE/POWER informs the system operator about this. In addition, it informs your program by way of return and feedback codes in the VSE/POWER-set user area of the XPCCB. VSE/POWER then waits for one of the following:

An order from your program (message or set-logical-destination).
A 'wait-for-order/signal' request from your program.
A command from the operator.
A selectable output queue entry to be queued.

Figure 42 on page 158 shows the sequence of the coding which you should provide in your program to cover the situation.

Instead of passing a wait-for-order/signal indication to VSE/POWER, your program may take either of the actions below.

- Give up the communication path (by an XPCC FUNC=DISCONN).

- Define or change one or more of the logical destination names for the device (by a set-logical-destination order), followed by another generic GET request.

Handle a Device-Setup Situation

Your program should analyse the verification SPL which VSE/POWER passes after the Get-service open request. As a result of this analysis, your program may have to initiate a device setup. The operational steps for this setup normally are as follows:

1. Your program passes a send-message order control record.

   This order instructs VSE/POWER to route the included message to the destination given in the order. VSE/POWER forwards the message to this destination, normally the operator responsible for the output device which is to be set up.

   Your order-control record may request VSE/POWER to hold a copy of the message in storage: the message may fail to reach its destination, and VSE/POWER's device-service task may therefore be operator bound. A copy of the message is helpful in this case; it enables the central operator to redisplay the message by means of a PDISPLAY M command. For more information about processing a send-message order, see the section "Send-Message Order" on page 191.

2. Your program waits for the reactivation of this output processing.

   The program does this by passing to VSE/POWER a wait-for-order/signal request (XPCCB bytes set as follows: PXUACT1 to PXUATWFR; PXUBTYP to zero).

When a device order or a signal gets queued for the communication path to your program, then VSE/POWER passes this order or signal.

```
       Coding in Your Program                  Comments
  _________________________________    _________________________________

       ... ... ...
  Open GET service                     This should be a generic GET-service
    XPCC FUNC=SENDR                    request.  VSE/POWER expects an (up-
      Check the return codes in        dated) SPL in your program's send
       register 15 and in the          buffer.
       XPCCB (byte IJBXRETC).
|   WAIT IJBXSECB                      A return SPL is in your program's
|     Check the VSE reason codes       reply buffer when the ECB is posted,
|      in the XPCCB byte IJBXREAS.     provided an eligible queue entry
|     Check the VSE/POWER return       was found.
       and feedback codes in XPCCB     The feedback code (byte PXPFBKCD)
       bytes PXPRETCD and PXPFBKCD,    is set to PXP04NOF if
       respectively.                   VSE/POWER cannot find a selectable
          |                            queue entry.  The remainder of the
          |                            sequence chart applies to this case.
          V
  Pass a wait-for-order/signal request
    XPCC FUNC=SENDR                    Pass a null buffer to VSE/POWER and
      Check the return codes as        be sure the wait-for-order/signal
|      shown above.                    flag (PXUACT1 set to PXUATWFR)
                                       is set in the XPCCB (see Note).
    WAIT IJBXSECB                      VSE/POWER has passed an order or a
      Check the VSE reason codes       signal when the ECB is posted.
       as shown above.
      Check the VSE/POWER return       VSE/POWER passes an output-arrived
       and feedback codes as shown     signal as soon as a selectable queue
       above.                          entry is queued.
      Analyze the order/signal
          |                            Let's assume that VSE/POWER did
          |                            pass the signal.  This means that
          |                            VSE/POWER is ready to accept a GET-
          |                            service request via the communica-
          |                            tion path.
          V
      To Part 2
```

Figure 42 (Part 1 of 2). Coding Sequence for a "No Entry Available" Situation

Coding in Your Program	Comments
┌── From Part 1 V	
Open GET service XPCC FUNC=SENDR	A retry of the originally passed generic GET-service request. Again, a selectable queue entry may not be available for processing. By the time VSE/POWER processes your program's request, the queued entry may have been manipulated from another source. Note: Since no selectable queue entry is available and no 'order pending' is indicated by VSE/POWER, your program should use the PXUATWFR request. This results in a VSE/POWER wait for the next order or signal, while a PXUATROR request would return immediate information about the availability of an order/signal.

Figure 42 (Part 2 of 2). Coding Sequence for a "No Entry Available" Situation

3. Output processing is reactivated.

   The operator issues a PGO command to indicate that the required setup work is done. This causes VSE/POWER to queue a reactivation-device order so that it can be passed to your program.

   Your program cannot reactivate output processing until VSE/POWER has passed a reactivate-device order. If VSE/POWER passes a device order other than reactivate (or setup), your program must respond to this order and reissue the wait-for-order/signal request.

Figure 43 shows the sequence of the coding which you should provide to cover the needs of a device setup and a reactivation of output processing. For more details about the processing of device orders, order-response records, and device signals, see the section "Process Order-Control Records and Signals" on page 176.

```
     Coding in Your Program                  Comments
 _________________________________     _________________________________

     ... ... ...
 Open GET service                      This should be a generic GET-service
   XPCC FUNC=SENDR                     request.  VSE/POWER expects an (up-
     Check the return codes in         dated) SPL in your program's send
      register 15 and in the           buffer.
      XPCCB (byte IJBXRETC).
   WAIT IJBXSECB                       A verification SPL is in your pro-
     Check the VSE reason codes        gram's reply buffer when the ECB is
      in the XPCCB byte IJBXREAS.      posted.
     Check the VSE/POWER return
      and feedback codes in XPCCB
      bytes PXPRETCD and PXPFBKCD,
      respectively.                    The remainder of the chart assumes
     Analyze the verification SPL.     that the indicated device character-
        |                              istics require a device setup.
        V
 Pass a send-message order             The order tells VSE/POWER where to
   XPCC FUNC=SENDR                     route the message which is part of
     Check the return codes as         the order-control record.
      shown above.
   WAIT IJBXSECB                       VSE/POWER's order response record
     Check the VSE reason codes        is in your program's reply buffer
      as shown above.                  when the ECB is posted.
     Check the VSE/POWER return
      and feedback codes as shown
      above.
        |
        V
     To Part 2
```

Figure 43 (Part 1 of 4). Coding Sequence for Device Setup and Reactivation

```
        Coding in Your Program                Comments
    ________________________________     ___________________________________

           ┌── From Part 1
           V
    Pass a wait-for-order/signal request
      XPCC FUNC=SENDR                    Pass a null buffer to VSE/POWER and
        Check the return codes as        be sure the "wait for order/signal"
        shown above.                     flag is set in the XPCCB.
           |
           V
      WAIT IJBXSECB                      VSE/POWER has passed an order or
        Check the VSE reason codes       a signal when the ECB is posted.
         as shown above.                 Let's assume that VSE/POWER passed
        Check the VSE/POWER return       a setup-device order.
         and feedback codes as
         shown above.
        Analyze the device order.        It indicates the number of pages
|          |                             the operator asks your program to
|          |                             retrieve from VSE/POWER and pass to
|          |                             the device for setup purposes.
           V
    Pass an order-response record
      XPCC FUNC=SENDR
        Check the return codes in
         register 15 and in the
         XPCCB (byte IJBXRETC).
      WAIT IJBXSECB                      VSE/POWER has processed the response
        Check the VSE reason codes       record and returned a null buffer
|        in the XPCCB byte IJBXREAS.     when the ECB is posted.
        Check the VSE/POWER return
         and feedback codes in XPCCB     Required programmed action:
         bytes PXPRETCD and PXPFBKCD,    1. Request VSE/POWER to pass the de-
         respectively.                      fined number of setup pages.
           |                             2. Reactivate normal processing when
           V                                the setup action is complete.
    GET spool data request  <────────┐
      XPCC FUNC=SENDR                |
        Check the return codes as    |
        shown above.                 |
      WAIT IJBXSECB                  |   When the ECB is posted, your pro-
        Check the VSE reason codes   |   gram's reply buffer is filled with
         as shown above.             |   spooled output records retrieved
        Check the VSE/POWER return   |   from the accessed queue entry.
         and feedback codes as       |
         shown above.                |
    Process the records in your re-  |   The data being passed may have to
     ply buffer.                     |   be replaced by strings of Xs.
     End of last setup page ┐; else ─┘
           ┌────────────────┘
           V
        To Part 3
```

Figure 43 (Part 2 of 4). Coding Sequence for Device Setup and Reactivation

```
      Coding in Your Program                      Comments
 ________________________________     ________________________________

       ┌── From Part 2
       V
Pass the setup-processed signal       Your program passes a null buffer
       |                              with PXUSIGNL of the XPCCB set to
       V                              PXUSSET.

  XPCC FUNC=SENDR                     This causes VSE/POWER to reset its
    Check the return codes as         retrieval pointers to the beginning
     shown above.                     of the queue entry being processed.
  WAIT IJBXSECB                       VSE/POWER has processed the signal
    Check the VSE reason codes        and returned a null buffer
     as shown above.                  when the ECB is posted.
    Check the VSE/POWER return
     and feedback codes as shown
     above.
       |
       V
Pass a wait-for-order/signal request
  XPCC FUNC=SENDR                     Your program passes a null buffer
    Check the return codes in         and the wait-for-order/signal flag
     register 15 and in the           in the XPCCB.
     XPCCB (byte IJBXRETC).
  WAIT IJBXSECB                       VSE/POWER has passed an order or a
    Check the VSE reason codes        signal when the ECB is posted.
     in the XPCCB byte IJBXREAS.
    Check the VSE/POWER return        Let's assume that VSE/POWER passed
     and feedback codes in XPCCB      a reactivate-device order.
     bytes PXPRETCD and PXPFBKCD,
     respectively.
    Analyze the order/signal
       |
       V
Pass an order-response record
  XPCC FUNC=SENDR
    Check the return codes as
    shown above.
  WAIT IJBXSECB                       VSE/POWER is ready to accept GET-
    Check the VSE reason codes        service requests and returned a
     as shown above.                  null buffer when the ECB is
    Check the VSE/POWER return        posted; VSE/POWER's return
     and feedback codes as shown      and feedback codes are OK.
     above.
       |
       V
    To Part 4
```

Figure 43 (Part 3 of 4). Coding Sequence for Device Setup and Reactivation

```
     Coding in Your Program                   Comments
 ________________________________     _________________________________
       ┌── From Part 3
       V
GET spool data request  <─────────┐
  XPCC FUNC=SENDR                 |
    Check the return codes as     |
    shown above.                  |
  WAIT IJBXSECB                   |   When the ECB is posted, your
    Check the VSE reason codes    |   program's reply buffer is filled
     as shown above.              |   with spooled output records re-
    Check the VSE/POWER return    |   trieved from the accessed queue
     and feedback codes as        |   entry.
     shown above.                 |
  Process the data passed by      |
   VSE/POWER.  If:                |
   More data is to be processed ──┘
   Else ┐
        |
        V
Pass a close request
  XPCC FUNC=SENDR                     Your program passes a null send buf-
    Check the return codes in         fer with the XPCCB bytes set as follows:
     register 15 and in the              PXUACT1 to PXUATRQS
     XPCCB (byte IJBXRETC).              PXUBTYP to zero
  WAIT IJBXSECB                       When the ECB is posted, VSE/POWER
    Check the VSE reason codes        has disposed of the just processed
     in the XPCCB byte IJBXREAS.      queue entry in accordance with the
    Check the VSE/POWER return        assigned disposition:
     and feedback codes in XPCCB         D - The entry is deleted
     bytes PXPRETCD and PXPFBKCD,        K - The entry's disposition is
     respectively.                           changed to L.
       |
       V
Process the next selectable
 queue entry or end the re-
 trieval of output.
```

Figure 43 (Part 4 of 4). Coding Sequence for Device Setup and Reactivation

Cancel Output Processing

Output processing is to be canceled when VSE/POWER receives a PFLUSH command for the device under your program's control. The command may request this cancelation with or without a HOLD specification.

For the PFLUSH command, VSE/POWER builds and queues a flush-device order. This order is passed to your program in response to a return-order/signal request.

Your program may delay the requested cancellation until a certain point in its processing; it may ignore the order by returning a not-accepted response. Normally, however, a subsystem would handle the device order as shown:

- In Figure 44 on page 165 for a PFLUSH without a HOLD specification.
- In Figure 45 on page 166 for a PFLUSH with a HOLD specification.

  If HOLD is specified, your program should continue output processing until a meaningful boundary (end of a page, for example) is reached. This may require your program to request a certain number of output records even after VSE/POWER passed the flush-device order. In addition, your program should request a checkpoint to be taken before it stops processing for the output that is to be canceled.

If a cancel message is to be written at the end of the canceled output, your program must build the message and write it to the device.

Request an Order or a Signal

VSE/POWER chains and passes device orders (or signals), using the first-in/first-out method. When it chains an order or signal, VSE/POWER indicates this by setting the user byte PXPINFO to PXPIORD. Your program should monitor the presence of a device order by testing this byte along with the VSE/POWER return and feedback codes.

For VSE/POWER to pass the order next in line, you must code the following in your program:

- If no order is queued and your program needs a certain order to continue -

  A wait-for-order/signal request. You do this by passing to VSE/POWER an XPCC FUNC=SENDR with a null send buffer and PXUACT1 set to PXUATWFR. You would use this method, for example, in a device-setup situation after your program has passed a send-message order.

- If an order is queued -

  A return-order/signal request. You do this by passing to VSE/POWER an XPCC FUNC=SENDR with a null send buffer and PXUACT1 set to PXUATROR.

Whenever VSE/POWER passes to you a device order, it expects you to return (in your send buffer) an order-response. For more information about the processing of orders, see the section "Process Order-Control Records and Signals" on page 176.

```
     Coding in Your Program                  Comments
 ____________________________________    ________________________________

                                         Assumption:  During GET data
                                         processing the PXPIORD
                                         (order signal queued) indication
                                         is set in the your program's XPCCB.
     ... ... ...
Pass a return-order/signal request
  XPCC FUNC=SENDR                        Your program passes a null buffer
    Check the return codes in            and the return-order/signal flag
     register 15 and in the              in the XPCCB.
     XPCCB (byte IJBXRETC).
  WAIT IJBXSECB                          VSE/POWER has passed an order or a
    Check the VSE reason codes           signal when the ECB is posted.
     in the XPCCB byte IJBXREAS.
    Check the VSE/POWER return
     and feedback codes in XPCCB
     bytes PXPRETCD and PXPFBKCD,
     respectively.
    Analyze the order/signal             Let's assume that VSE/POWER passed
       |                                 the device order for a PFLUSH with-
       V                                 out HOLD.
Pass an order-response record
  XPCC FUNC=SENDR
    Check the return codes as
    shown above.
  WAIT IJBXSECB                          VSE/POWER has processed the response
    Check the VSE reason codes           record and returned a null buffer
     as shown above.                     when the ECB is posted.
    Check the VSE/POWER return
     and feedback codes as shown
     above.
       |
       V
Pass a close request
  XPCC FUNC=SENDR                        Your program passes a null send buffer
    Check the return codes as            with the XPCCB bytes set as follows:
    shown above.                            PXUACT1 to PXUATRQS
                                            PXUBTYP to zero
  WAIT IJBXSECB                          VSE/POWER has processed the request
    Check the VSE reason codes           and returned a null buffer when the
    as shown above.                      ECB is posted.  VSE/POWER
                                         deletes the currently processed
    Check the VSE/POWER return           queue entry if the entry's disposi-
     and feedback codes as shown         tion was D.  VSE/POWER retains the
     above.                              entry, with a disposition of L, if
       |                                 its original disposition was K.
       V
Get-service request for the
 next selectable queue entry
     ... ... ...
```

Figure 44. Coding Sequence for a PFLUSH without HOLD

```
      Coding in Your Program                      Comments
 ________________________________       ________________________________

                                         Assumption:  During GET data
                                         processing the PXPIORD
                                         (order signal queued) indication
                                         is set in your program's XPCCB.
     ... ... ...
 Pass a return-order/signal request
   XPCC FUNC=SENDR                       Your program passes a null buffer
     Check the return codes in           and the return-order/signal flag
      register 15 and in the             in the XPCCB.
      XPCCB (byte IJBXRETC).
   WAIT IJBXSECB                         VSE/POWER has passed an order or a
     Check the VSE reason codes          signal when the ECB is posted.
      in the XPCCB byte IJBXREAS.           Let's assume that VSE/POWER
     Check the VSE/POWER return          passed a device order for a PFLUSH
      and feedback codes in XPCCB        with HOLD.
      bytes PXPRETCD and PXPFBKCD,
      respectively.
     Analyze the order/signal
        |
        V
 Pass an order-response record
   XPCC FUNC=SENDR
     Check the return codes as
      shown above.
   WAIT IJBXSECB                         VSE/POWER has processed the response
     Check the VSE reason codes          record and returned a null buffer
      as shown above.                    when the ECB is posted.
     Check the VSE/POWER return and
      feedback codes as shown above.
        |
        V
 GET spool data request  <----------┐
   XPCC FUNC=SENDR                  |
     Check the return codes as      |
      shown above.                  |
   WAIT IJBXSECB                    |    When the ECB is posted, your
     Check the VSE reason codes     |    program's reply buffer is filled
      as shown above.               |    with spooled output records re-
     Check the VSE/POWER return     |    trieved from the accessed queue
      and feedback codes as         |    entry.
      shown above.                  |
     If the end of the current      |
      page is reached ┐; else ------┘
        ┌─────────────┘
        |
        V
     To Part 2
```

Figure 45 (Part 1 of 2). Coding Sequence for a PFLUSH with HOLD

```
   Coding in Your Program                Comments
 ______________________________    ________________________________

      ┌── From Part 1
      |
      V
Pass a checkpoint request
  XPCC FUNC=SENDR                   Your program passes a checkpoint-
    Check the return codes in       control record in its send buffer.
     register 15 and in the
     XPCCB (byte IJBXRETC).
  WAIT IJBXSECB                     When the ECB is posted, VSE/POWER
    Check the VSE reason codes      has passed a checkpoint-response
     in the XPCCB byte IJBXREAS.    record to your program's reply buf-
    Check the VSE/POWER return      fer.
     and feedback codes in XPCCB
     bytes PXPRETCD and PXPFBKCD,
     respectively.
      |
      V
Pass a flush-hold request
  XPCC FUNC=SENDR                   Your program passes a null send buf-
    Check the return codes as       fer and XPCCB bytes set as follows:
     shown above.                      PXUACT1 set to PXUATFLH
                                       PXUBTYP set to zero
  WAIT IJBXSECB                     VSE/POWER has processed the request
    Check the VSE reason codes      and returned a null buffer
     as shown above.                when the ECB is posted; process-
    Check the VSE/POWER return      ing of the affected output queue
     and feedback codes as shown    entry by VSE/POWER is canceled.
     above.                            The complete output queue entry
      |                             is retained in its output queue with
      |                             the class and priority assignments
      |                             unchanged.  The queue entry's dis-
      |                             position, however, is changed to:
      |                                H if it was D.
      V                                L if it was K.
Get-service request for
 the next selectable
 queue entry.
     ... ... ...
```

Figure 45 (Part 2 of 2). Coding Sequence for a PFLUSH with HOLD

STOP THE DEVICE

Normally, the stopping of a device is triggered by VSE/POWER when it processes a PSTOP DEV command for the device or a PEND command.

Either command causes VSE/POWER to build a stop-device order and to add this order to the order chain for the device. The order may request the device to be stopped:

- At the end of the currently processed output

  A PSTOP command with EOJ or a PEND command was issued. Your program must provide for continued processing of output until the end of the currently processed output is reached. Figure 46 on page 169 shows the coding sequence that should be followed.

- At once for restart at the point of interruption

  A PSTOP command with RESTART was issued. Your program must provide for continued processing of output until the end of a logical boundary (a page for a printer, for example) is reached. At this point, have your program request a checkpoint because setting up output processing on restart for the queue entry is your program's responsibility. Figure 47 on page 171 shows the coding sequence that should be followed.

- At once for restart from the beginning

  Neither EOJ nor RESTART was specified in the PSTOP command. In this case, your program should:

  1. Purge the data that may be contained in a device buffer, if any.

  2. Issue a quit request.

```
    Coding in Your Program                 Comments
 ____________________________      ____________________________

                                   Assumption:  During GET data
                                   processing the PXPIORD
                                   (order signal queued) indication
                                   is set in your program's XPCCB.
     ... ... ...
 Pass a return-order/signal request
   XPCC FUNC=SENDR                 Your program passes a null buffer
     Check the return codes in     and the return-order/signal flag
      register 15 and in the       in the XPCCB.
      XPCCB (byte IJBXRETC).
   WAIT IJBXSECB                   VSE/POWER has passed an order or a
     Check the VSE reason codes    signal when the ECB is posted.
      in the XPCCB byte IJBXREAS.     Let's assume that VSE/POWER
     Check the VSE/POWER return    passed a device order for a PSTOP
      and feedback codes in XPCCB  with EOJ.
      bytes PXPRETCD and PXPFBKCD,
      respectively.
     Analyze the order/signal
        |
        V
 Pass an order-response record
   XPCC FUNC=SENDR
     Check the return codes as
      shown above.
   WAIT IJBXSECB                   VSE/POWER has processed the response
     Check the VSE reason codes    record and returned a null buffer
      as shown above.              when the ECB is posted.
     Check the VSE/POWER return and
      feedback codes as shown above.
        |
        V
 GET spool data request  <-------┐
   XPCC FUNC=SENDR               |
     Check the return codes as   |
      shown above.               |
   WAIT IJBXSECB                 |  When the ECB is posted, your
     Check the VSE reason codes  |  program's reply buffer is filled
      as shown above.            |  with output records retrieved
     Check the VSE/POWER return  |  from the accessed queue entry.
      and feedback codes as      |
      shown above.               |
     If the end of the queue     |
      entry is reached ┐; else --┘
        ┌──────────────┘
        |
        V
     To Part 2
```

Figure 46 (Part 1 of 2). Coding Sequence, Device Stop after End of Output

```
      Coding in Your Program                     Comments
 ________________________________    ________________________________

       ┌─── From Part 1
       |
       V
 Empty hardware I/O buffers          Applies if the device is buffered or
       |                             connected via a communication link.
       |                                Your program must ensure that re-
       |                             cords still in a hardware buffer are
       |                             actually written to the device before
       |                             the retrieval service for the output
       |                             is closed. This avoids that VSE/POWER
       |                             deletes the output before all of the
       |                             output records have been transferred
       V                             to and processed by the device.
 Issue a Close request
   XPCC FUNC=SENDR                   Your program passes a null buffer
     Check the return codes in       with the XPCCB byte
      register 15 and in the            PXUBTYP set to zero
      XPCCB (byte IJBXRETC).            PXUACT1 equated to PXUATRQS
       |                             When the ECB is posted, VSE/POWER:
   WAIT IJBXSECB                     -  Has returned a null buffer.
     Check the VSE reason codes      -  Has deleted the output if this
      in the XPCCB byte IJBXREAS.       this output's disposition was D.
     Check the VSE/POWER return      -  Has changed the output's dis-
      and feedback codes in XPCCB       position to L if this disposition
      bytes PXPRETCD and PXPFBKCD,      was K.
      respectively.
       |
       V
 Pass a device-stopped signal        Your program passes a null send buf-
   XPCC FUNC=SENDR                   fer with XPCCB bytes set as follows:
     Check the return codes as          PXUSIGNL to PXUSDSTP
      shown above.                      PXUBTYP to zero
       |                             VSE/POWER has processed the signal
   WAIT IJBXSECB                     and returned a null buffer
     Check the VSE reason codes      when the ECB is posted.
      as shown above.                   VSE/POWER informs about the de-
     Check the VSE/POWER return      vice-stopped condition by a message
      and feedback codes in XPCCB    to the PSTART device operator and to
      bytes PXPRETCD and PXPFBKCD,   the user who issued the PSTOP
      respectively.                  (or PEND) command, thus disconnecting
       |                             the communication path.
       V
 Give up the communication path
   XPCC FUNC=DISCPRG                 The communication path is removed.
     Check the return codes as
      shown above.
       |
       V
     ... ... ...
```

Figure 46 (Part 2 of 2). Coding Sequence, Device Stop after End of Output

This chart shows only how a stop with a restart possibility differs from a stop after end of job.

```
      Coding in Your Program                       Comments
 ________________________________    ________________________________

     ... ... ...
 Pass the required order-response
 record
   XPCC FUNC=SENDR
     Check the return codes in
      register 15 and in the
      XPCCB (byte IJBXRETC).
   WAIT IJBXSECB                     VSE/POWER has processed the response
     Check the VSE reason codes      record and returned a null buffer
      in the XPCCB byte IJBXREAS.    when the ECB is posted.
     Check the VSE/POWER return
      and feedback codes in XPCCB
      bytes PXPRETCD and PXPFBKCD,
      respectively.
        |
        V
 GET spool data request  <---------┐
   XPCC FUNC=SENDR                 |
     Check the return codes as     |
      shown above.                 |
   WAIT IJBXSECB                   | When the ECB is posted, your
     Check the VSE reason codes    | program's reply buffer is filled
      as shown above.              | with output records retrieved
     Check the VSE/POWER return    | from the accessed queue entry.
      and feedback codes as        |
      shown above.                 |
     If the end of a logical       |
      boundary is reached ┐; else -┘
        ┌─────────────────┘
        |
        V
     To Part 2
```

Figure 47 (Part 1 of 2). Coding Sequence, Device Stop with a Restart Possibility

```
     Coding in Your Program                   Comments
 ________________________________      ________________________________

        ┌── From Part 1
        |
        V
 Pass a checkpoint request
   XPCC FUNC=SENDR                     Your program passes a checkpoint-
     Check the return codes in         control record in its send buffer.
      register 15 and in the
      XPCCB (byte IJBXRETC).
   WAIT IJBXSECB                       When the ECB is posted, VSE/POWER
     Check the VSE reason codes        has passed a checkpoint-response
      in the XPCCB byte IJBXREAS.      record to your program's reply buf-
     Check the VSE/POWER return        fer.
      and feedback codes in XPCCB
      bytes PXPRETCD and PXPFBKCD,
      respectively.
        |
        V
 Empty hardware I/O buffers            This is the same as for a termina-
        |                              tion after end of job; see the
        |                              coding sequence shown in the preced-
        V                              ing illustration.
 Pass a quit request
   XPCC FUNC=SENDR                     Your program passes a null send buf-
     Check the return codes as         fer and XPCCB bytes set as follows:
      shown above.                        PXUACT1 set to PXUATABR
        |                                 PXUBTYP set to zero
        V
   WAIT IJBXSECB                       VSE/POWER has processed the request
     Check the VSE reason codes        when the ECB is posted.  The queue
      as shown above.                  entry being processed is retained
     Check the VSE/POWER return        by VSE/POWER with unchanged priority
      and feedback codes as            and disposition assignments.
      shown above.
        |
        V
 Pass a device-stopped signal          This and the remainder of the coding
                                       sequence is the same as for a term-
                                       ination after end of job (see the
                                       preceding illustration).
```

Figure 47 (Part 2 of 2). Coding Sequence, Device Stop with a Restart Possibility

HANDLE AN ABNORMAL-END SITUATION

An abnormal-end situation may arise because of an error condition that occurs during the processing of an output queue entry. Such a situation may also occur because of an error condition within VSE/POWER. This section briefly discusses the two kinds of abnormal-end situations.

Output-Related Abnormal End

This type of an abnormal-end situation may be determined by VSE/POWER or by your program:

- Abnormal-end situation determined by VSE/POWER: VSE/POWER removes the communication path immediately. It writes a message to the system operator and to the device owner to inform about this abnormal end of output processing. VSE/POWER retains the currently processed queue entry in the entry's output queue with priority and disposition assignments unchanged. It performs the required accounting.
- Abnormal-end situation determined by your program: Your program should analyze the situation and then do one of the following:
  - Cancel itself. This action is indicated if there is no chance for continued useful work. If this occurs, VSE/POWER is informed about it by the XPCC interface.
  - Remove the communication path by an XPCC request specifying FUNC=DISCPRG. This action is indicated if there is no chance for continued useful processing of data passed via the communication path to or from your program.

  In either of the above cases, VSE/POWER's action is the same as for an abnormal-end situation determined by VSE/POWER, which is discussed in the next section.
  - Remove the communication path by an XPCC request specifying FUNC=DISCONN when the last FUNC=SENDR request has been completed (SECB posted). This action is indicated if, for example, your program can no longer write to the output device.

    Your program should inform the system operator and, if possible, also the device owner of the type of failure. If the device was active when the failure occurred, have your program save a checkpoint, a VSE/POWER-assigned record number lower than the number of the failing record. Your program can use this record number as a restart point when processing of the interrupted queue entry is resumed.
- Output processing failure for a 'protected' queue entry: See "Abnormal-End Condition During GET" on page 57.

Figure 48 on page 174 shows the coding sequence that should be followed.

```
     Coding in Your Program                     Comments
 ___________________________________    __________________________________

     ... ... ...
 Pass a send-message order
   XPCC FUNC=SENDR                      Tells VSE/POWER where to route the
     Check the return codes in          message which is part of the order-
      register 15 and in the            control record.
      XPCCB (byte IJBXRETC).
   WAIT IJBXSECB                        VSE/POWER's order response record is
     Check the VSE reason codes         in your program's reply buffer when
      in the XPCCB byte IJBXREAS.       the ECB is posted.
     Check the VSE/POWER return
      and feedback codes in XPCCB
      bytes PXPRETCD and PXPFBKCD,
      respectively.
       |
       V
 Pass a checkpoint request
   XPCC FUNC=SENDR                      Your program passes a checkpoint-
     Check the return codes as          control record in its send buffer.
      shown above.
   WAIT IJBXSECB                        When the ECB is posted, VSE/POWER
     Check the VSE reason codes         has passed a checkpoint-response
      as shown above.                   record to your program's reply buf-
     Check the VSE/POWER return and     fer.
      feedback codes as shown above.
       |
       V
 Pass a quit request
   XPCC FUNC=SENDR                      Your program passes a null send buf-
     Check the return codes as          fer and XPCCB bytes set as follows:
     shown above.                          PXUACT1 set to PXUATABR
       |                                   PXUBTYP set to zero
       V
   WAIT IJBXSECB                        VSE/POWER has processed the request
     Check the VSE reason codes         when the ECB is posted.  The inter-
      as shown above.                   rupted queue entry is retained by
     Check the VSE/POWER return         VSE/POWER with unchanged priority
      and feedback codes as             and disposition assignments.
      shown above.
       |
       V
 Give up the communication path
   XPCC FUNC=DISCONN
     Check the return codes in
      register 15 and in the
      XPCCB (byte IJBXRETC).            The communication path is removed.
       |
       V
     ... ... ...
```

Figure 48. Coding Sequence, Abnormal End Because of a Device Failure

- Output-processing failure indicated by your program

  You can issue a 'quit-and-lock' request at any point during the retrieval of a queue entry. The request causes VSE/POWER to requeue the currently processed queue entry in the appropriate class chain with a temporary disposition of Y for the purpose of:

  - Indicating that a problem has occurred during output processing, and

  - Preventing that the output queue entry is handled again until the subsystem has taken some action (for example, issued the PALTER command to alter the temporary disposition to a dispatchable one).

  A queue entry with disposition Y is not automatically processed by the various VSE/POWER tasks. Your program can make use of the CTL service to

  - Get a display of all queue entries that have a disposition of Y by entering the PDISPLAY ALL,CDISP=Y command, and

  - Alter this disposition for a queue entry to make it eligible for processing again. To reset disposition Y of a queue entry to its original one, use the PALTER queue,jobname,DISP=* command.

  For further information on disposition refer to the VSE/POWER Installation and Operations Guide.

Abnormal End of VSE/POWER

VSE/POWER itself may happen to be canceled during output processing. The XPCC interface informs your program about this by passing to your program XPCCB return or reason codes of IJBXNOC3 and IJBXABDC, respectively. Your program can, in this case:

1. Empty hardware-output buffers, if any.

2. When VSE/POWER is up again, restart the interrupted processing either:

   - At a suitable checkpoint (if the output was checkpointed). Obtaining checkpoints during data retrieval is described under "Request a Checkpoint" on page 49; restarting at a checkpoint is discussed in the section "Request a Restart" on page 50.
   - At the beginning of the interrupted output.

For more information about the retrieval and restart of a queue entry, see "Request a GET Service" on page 41.

If VSE/POWER or the XPCC interface happens to be canceled while processing a 'protected' output queue entry, VSE/POWER recovery (at system warm start) or the VSE/POWER device-service task will requeue the output entry with disposition Y to the non-dispatchable queue. For creation of a protected queue entry see "Abnormal-End Condition During GET" on page 57.

PROCESS ORDER-CONTROL RECORDS AND SIGNALS

Orders and signals are used to synchronize a VSE/POWER device-service task with your program.

An order is a control record which is passed from one side of a communication path to the other. A signal is a status indication that is passed to the other end of the communication path. Orders that VSE/POWER can pass to your program are referred to as device orders; orders that your program can pass to VSE/POWER are called subsystem orders.

VSE/POWER-Built Device Orders: VSE/POWER builds a device order whenever it processes any of the following commands for a device under your program's control:

Command	Order-Type
PSTART	Start-device order
PSTOP	Stop-device order
PRESTART	Restart-device order
PGO	Reactivate-device order
PSETUP	Setup-device order
PFLUSH	Cancel-output device order
PXMIT	Transmit-command device order

VSE/POWER handles device orders in a first-in first-out way by chaining them, one behind the other, separately for each device controlled by your program. VSE/POWER accepts a command for a device even after a PSTOP DEV command was processed for this device, that is, until your program has passed a device-stop signal.

Subsystem-Originated Orders: The subsystem (your program) would build an order and pass it to VSE/POWER whenever the need arises. Your program can build and pass orders of the following type:

Send-message order.
Set-logical-destination order.

Process a Device Order

Process Overview: When having passed a device order to your program, VSE/POWER expects that the program analyses the order immediately and returns a corresponding order-response record. If your program fails to return this record, VSE/POWER discontinues the communication path and informs your program by a return code of PXPRCPVL together with the applicable feedback code. VSE/POWER discontinues the communication path also if your program's order-response record does not correspond to the type of order passed by VSE/POWER.

The order-response record indicates your program's decision: accepted or not accepted. If the decision is not accepted, the record may also indicate a reason for rejecting the device order; it may include a message for VSE/POWER to route to the user whose command triggered the the device order. For the programmed actions that are to be coded in order to return an order-response record, refer to Figure 49 on page 178.

A message generated by VSE/POWER in response to a command is routed to the command originator whose nodeid and userid may be derived from the device-order header. A message passed to VSE/POWER as part of an order-response control record is routed to the user indicated in this record; by default, this is the originator of the command. For details on message routing, refer to Figure 51 on page 182.

A device order, once accepted by an order-response record, may be processed by your program some time later. For example, after having accepted an immediate-stop device order, your program can request a checkpoint to be taken before it processes the order. There is one exception, however: the start-device order. Your program must process this order immediately and return the result of this processing by way of an order-response record valid for this device order.

Sequence of Events

1. VSE/POWER chains a device order for being passed via a communication path when it processes a command for the involved device. This may occur at any time. VSE/POWER indicates the chaining of a device order.

2. VSE/POWER indicates the chaining of a device order by setting the order-pending flag in the XPCCB for the communication path. When this XPCCB is passed to the other end (your program), VSE/POWER expects, sooner or later, a return-order/signal request to be returned. In short, your program should be ready to pick up and analyze a device order each time after VSE/POWER has passed to your program a block of output records.

3. In response to a return-order/signal request, VSE/POWER passes the device order at the head of the chain if two or more such orders are chained for the communication path. The order-pending flag remains

set as long as a device order waits for being passed to your program.

Figure 49 shows the coding sequence which you should follow in your program for the handling of device orders.

Size of Your Reply Buffer: An order-control record may have a length of up to 180 bytes. Therefore, the size of your program's reply buffer should be 180 bytes or larger.

```
      Coding in Your Program                    Comments
 ________________________________    _____________________________

      ... ... ...
 GET spool data request for the
 next block
   XPCC FUNC=SENDR
     Check the return codes in
      register 15 and in the
      XPCCB (byte IJBXRETC).
   WAIT IJBXSECB                     When the ECB is posted, your
     Check the VSE reason codes      program's reply buffer is filled
      in the XPCCB byte IJBXREAS.    with spooled output records re-
     Check the return and feedback   trieved from the accessed queue
      codes in XPCCB bytes PXPRETCD  entry.
      and PXPFBKCD, respectively.
     Check the XPCCB byte PXPINFO    An order or signal is chained if the
        |                            PXPIORD bit of this byte is set on.
        V
     Process the data passed by      Prepare this data for writing it
      VSE/POWER                      to the involved output device.
        |
        V
 Pass a return-order/signal request
   XPCC FUNC=SENDR                   Your program passes a null buffer
     Check the return codes as       with XPCCB bytes set as follows:
      shown above.                      PXUBTYP to zero
        |                               PXUACT1 to PXUATROR
        V
   WAIT IJBXSECB                     When the ECB is posted, VSE/POWER
     Check the VSE reason codes      has passed an order or a signal, if
      as shown above.                there was one; if there was none,
     Check the VSE/POWER return and  VSE/POWER indicates this by a re-
      feedback codes as shown above. turn- and feedback-code combination
     Analyze the order               of PXPRCOKF and PXP04NOQ.
        |                               Let's assume that VSE/POWER
        V                            passed a device order.
     To Part 2
```

Figure 49 (Part 1 of 2). Coding Sequence, Processing of Device Orders

```
      Coding in Your Program                       Comments
 _________________________________       ________________________________

        ┌── From Part 1
        V
 Pass the required order-response
 record
   XPCC FUNC=SENDR                        With only the control record in
     Check the return codes in            your program's send buffer and with
      register 15 and in the              the XPCCB's byte PXUBTYP set to
      XPCCB (byte IJBXRETC).              PXUBTCTL.
   WAIT IJBXSECB                          VSE/POWER has processed the response
     Check the VSE reason codes           record and returned a null buffer
      in the XPCCB byte IJBXREAS.         when the ECB is posted.
     Check the return and feedback
      codes in XPCCB bytes PXPRETCD
      and PXPFBKCD, respectively.
        |
        V
```

Figure 49 (Part 2 of 2). Coding Sequence, Processing of Device Orders

Process a Subsystem Order

To pass an order to VSE/POWER, your program must:

1. Set up the device order as the only data in the communication path's send buffer.

2. Issue an XPCC request specifying FUNC=SENDR. The XPCCB used for the request must have its user-information byte PXUBTYP set to PXUBTCTL.

VSE/POWER analyzes the order and returns to your program the corresponding order-response record. For information about the format and contents of the orders and response records, see "Device/Subsystem Orders and Order Response Records" below.

Device/Subsystem Orders and Order Response Records

Device/subsystem orders and order-response records are similar in format. Both types of control records have a header section and a variable-data section. Following below are:

1. The format and description of the header section of a device/subsystem order. The description includes a general discussion of the data section; the required details about order data sections are given separately by device/subsystem orders.

2. The format and description of the order-response record, including its data section.

Device/Subsystem-Order Header Section

For the format of this record section and a discussion of its contents, refer to Figure 50. In the assembly output listing for the PWRSPL macro with TYPE=MAP, you find a DSECT for the record section at the label PORDER.

Bytes	Field Name	Contents / Description
0 - 1	PORDRLEN	Record length (in binary notation).
2	PORDTYPE	X'05' – Device-order indicator.
3	PORDMOD	Device-order type:

Symbol	Value	Order-Type	Triggered By
PORDMSTR	X'01'	Start device	PSTART
PORDMSTP	X'02'	Stop device	PSTOP
PORDMRST	X'03'	Restart device	PRESTART
PORDMPGO	X'04'	Reactivate device	PGO
PORDMSET	X'05'	Setup device	PSETUP
PORDMFLH	X'06'	Cancel processing	PFLUSH
PORDMXMT	X'07'	User defined	PXMIT
PORDMSND	X'10'	Send message	Subsystem
PORDMSLD	X'11'	Set logical destination	Subsystem

Figure 50 (Part 1 of 2). Format of the Device/Subsystem-Order Header Section

Bytes	Field Name	Contents / Description
4	PORDFLAG	Flag byte – to be set to X'80' by the subsystem in a send-message order if the message is to be held for redisplay (by a PDISPLAY M command).
5	PORDMSGL	Length of message (in binary notation) – To be supplied by the subsystem in a send-message order.
6 - 7		Reserved.
8 - F	PORDSUBS	Requesting subsystem's name (in character notation).
10 - 17	PORDNODE	Requesting node's name (in character notation). Your own VSE system's node name (or blank) if the triggering command was submitted within the domain of your node.
18 - 1F	PORDUSER	Requesting user's identifier (in character notation). Blank if the triggering command was entered by a central operator.
20 - n		Variable-data area – See also "Note" below.

Note: Details are given in the sections discussing the device/subsystem orders. The variable-data area includes a parameter string if one was specified in the triggering command. This string normally provides operator-specified information that your program needs. Tell your operator what to specify and how.

VSE/POWER's requirements regarding the parameter string are:

- It may not be longer than 60 characters. This includes blanks or commas that your program may need as delimiters.
- It must start with an alphameric character in the first character position.
- It must include at least one blank in any of the second through 16th character positions.
- An apostrophe (') within the string must be entered by the operator as two apostrophes ('').

Figure 50 (Part 2 of 2). Format of the Device/Subsystem-Order Header Section

Order-Response Record

When VSE/POWER passes a device order, it expects your program to return the corresponding order-response record with your program's next XPCC request. If your program passes an invalid response record, VSE/POWER:

1. Rejects this record with a return/feedback-code combination of PXPRCERR/PXPO8UXR in the XPCCB bytes PXPRETCD and PXPFBKCD.

2. Waits for a new corrected response record.

When your program passes a subsystem order, VSE/POWER returns the corresponding order-response record also in response to the next XPCC request.

For the format of the record and a discussion of its contents refer to Figure 51. In the assembly output listing for the PWRSPL macro with TYPE=MAP, you find a DSECT for the record section at the label PORDRESP.

Bytes	Field Name	Contents / Description
0 - 1	PORSRLEN	Record length (in binary notation).
2	PORSTYPE	X'06' – Order-response record indicator.
3	PORSMOD	Device-order type – The type indicator of the device order to which a response is being made. Consider picking up field PORDMOD of the device order, which is discussed under "Device/Subsystem-Order Header Section" on page 180.
4		Reserved.
5	PORSMSGL	Length of the message (in binary notation), if there is one; else X'00'.
6	PORSRETC	Order return code:

Symbol	Value	Explanation
PORSOK	X'00'	Order accepted.
PORSINV	X'08'	Order not accepted.

Figure 51 (Part 1 of 2). Format of the Order-Response Control Record

Bytes	Field Name	Contents / Description
7	PORSFDBK	Order feedback code: Symbol — Value — Explanation From the subsystem to VSE/POWER: PORSFOK X'00' All OK. PORSFPAR X'01' Missing or invalid parameter string. PORSFONA X'02' Subsystem-internal reason. PORSFDUN X'03' Device to be started is unknown. PORSFDBS X'04' Device to be started is busy. PORSFDOS X'05' Device to be started is out of service. PORSFDRJ X'06' Device start rejected for subsystem-internal reason. From VSE/POWER to the subsystem: PORSFINV X'01' Order is invalid or unknown. PORSFOTS X'02' Order is too short. PORSFMSG X'03' Message text is too long. PORSFSLD X'04' Invalid destination in a preceding set-logical destination order.
8 - F	PORDSUBS	Destination subsystem's name (in character notation).
10 - 17	PORDNODE	Destination node's name (in character notation). Blank if the message passed with the order-response record is to be routed to the system operator.
18 - 1F	PORDUSER	Destination user's identifier (in character notation). Blank if the message passed with the order-response record is to be routed to the system operator.
20 - 97	PORSMSG	Message text - See "Note" below.

Note: Applies to order-response records from the subsystem to VSE/POWER.

The contents of this field are picked up by VSE/POWER. These contents, the message text, must be alphameric; it can be up to 120 characters long. A shorter text must be padded with blanks at its end. Your program can include an error message here if, for example, the parameter string passed with the device order is in error. VSE/POWER routes this message to the user identified by fields PORDSUBS, PORDNODE, PORDUSER, and translates the message text to uppercase.

Figure 51 (Part 2 of 2). Format of the Order-Response Control Record

Start-Device Order

VSE/POWER passes the order to your program when a PSTART DEV command is processed for a device under your program's control. Not until it has accepted the order (by a corresponding order-response record) can your program request VSE/POWER to pass output spooled for the device.

If the device cannot be started, your program must indicate this and give a reason by setting the return and feedback codes in the order-response record. A return code other than PORSROK (X'00') causes VSE/POWER to discontinue the communication path.

Figure 52 shows the format of the device order's data section and the return and feedback codes that your program may have to supply in the response record.

Device-Order Data Section:

Bytes	Field Name	Contents / Description
20 - 27	PORDSDEV	Device name specified in the PSTART command.
28 - 2B	PORDSCLS	Class(es) specified in the PSTART command.
2C - 2E		Reserved.
2F	PORDSPSL	Length of parameter string (in binary notation).
30 - 6B	-	Parameter string as supplied in the PSTART command.

Response-Record Return and Feedback Codes:

Return Code		Feedback Code		
Symbol	Value	Symbol	Value	Explanation
PORSROK	X'00'	PORSFOK	X'00'	Order accepted, device started.
PORSRINV	X'08'			Order not accepted.
		PORSFPAR	X'01'	Parameter string is missing or invalid.
		PORSFONA	X'02'	Subsystem-internal reason.
		PORSFDUN	X'03'	Device to be started is unknown.
		PORSFDBS	X'04'	Device to be started is busy.
		PORSFDOS	X'05'	Device to be started is out of service.
		PORSFDRJ	X'06'	Device start rejected for subsystem-internal reason.

Figure 52. Data Section and Return and Feedback Codes for a Start-Device Order

Stop-Device Order

Figure 53 shows the format of the device order's data section and the return and feedback codes that your program may have to supply in the response record.

VSE/POWER passes the order to your program when either of the following occurs:

- A PSTOP DEV command is processed for a device under your program's control.
- An orderly VSE/POWER shutdown in response to a PEND command is in process.

VSE/POWER honors your program's GET-spooled data requests even after the program has passed the corresponding response record. In fact, it honors these requests until your program has passed its device-stopped signal.

Device-Order Data Section:

Bytes	Field Name	Contents / Description		
20	PORDPTRB	Termination request byte:		
		Symbol	Value	Explanation
		PORDPEOJ	X'80'	Stop at end of job
		PORDPIMM	X'40'	Stop immediately
		PORDPRST	X'20'	Stop for later restart
21 - 22		Reserved.		
23	PORDPPSL	Length of parameter string		
24 - 5F	PORDPPRM	Parameter string		

Response-Record Return and Feedback Codes:

Return Code		Feedback Code		
Symbol	Value	Symbol	Value	Explanation
PORSROK	X'00'	PORSFOK	X'00'	Order accepted.
PORSRINV	X'08'			Order not accepted.
		PORSFPAR	X'01'	Parameter string is missing or invalid.
		PORSFONA	X'02'	Subsystem-internal reason.

Figure 53. Data Section and Return and Feedback Codes for a Stop-Device Order

A PSTOP DEV,..,FORCE command does not cause a stop-device order to be passed by VSE/POWER. Instead, VSE/POWER discontinues the communication path immediately. VSE/POWER informs your program about this by a return- and feedback-code combination of PXPRCNOC and PXP1OPSP.

Setup-Device Order

VSE/POWER passes the order to your program when a PSETUP DEV command is processed for a device under your program's control. The order indicates the number of pages that are to be printed so that the operator can do the required device setup. As a help for the device operator, consider having your program replace on the setup pages:

All letters by the character X.
Each digit of a number by a 9.

Figure 54 below shows the format of the device order's data section and the return and feedback codes that your program may have to supply in the response record.

Device-Order Data Section:

Bytes	Field Name	Contents / Description
20 - 23	PORDUPGE	Number of pages (in binary notation).
24 - 2E		Reserved.
2F	PORDUPSL	Length of parameter string (in binary notation).
30 - 6B	PORDUPRM	Parameter string

Response-Record Return and Feedback Codes:

Return Code		Feedback Code		
Symbol	Value	Symbol	Value	Explanation
PORSROK	X'00'	PORSFOK	X'00'	Order accepted.
PORSRINV	X'08'			Order not accepted.
		PORSFPAR	X'01'	Parameter string is missing or invalid.
		PORSFONA	X'02'	Subsystem-internal reason.

Figure 54. Data Section and Return and Feedback Codes for a Setup-Device Order

Your program must inform VSE/POWER when it is finished with the setup processing. This is done by passing a setup-processed signal. The signal causes VSE/POWER to re-positions its retrieval pointers to the beginning of the currently processed queue entry.

Reactivate-Device Order

VSE/POWER passes the order to your program when a PGO DEV command is processed for a device under your program's control. Figure 55 above shows the format of the device order's data section and the return and feedback codes that your program may have to supply in the response record.

Restart Device Order

VSE/POWER passes the order to your program when a PRESTART DEV command is processed for a device under your program's control. Figure 56 on page 188 shows the format of the device order's data section and the return and feedback codes that your program may have to supply in the response record.

Device-Order Data Section:

Bytes	Field Name	Contents / Description
20 - 22		Reserved.
23	PORDGPSL	Length of parameter string
24 - 5F	PORDGPRM	Parameter string

Response-Record Return and Feedback Codes:

Return Code		Feedback Code		
Symbol	Value	Symbol	Value	Explanation
PORSROK	X'00'	PORSFOK	X'00'	Order accepted.
PORSRINV	X'08'			Order not accepted.
		PORSFPAR	X'01'	Parameter string is missing or invalid.
		PORSFONA	X'02'	Subsystem-internal reason.

Figure 55. Data Section and Return and Feedback Codes for a Reactivate-Device Order

Device-Order Data Section:

Bytes	Field Name	Contents / Description
20	PORDTFLG	Restart-sign flag:
21 - 23		Reserved.
24 - 27	PORDTPGE	Number of pages/printlines – How to interpret this number depends on your application.
28 - 2E		Reserved.
2F	PORDTPSL	Length of parameter string
30 - 6B	PORDTPRM	Parameter string

Symbol	Value	Explanation
PORDTPOS	X'80'	Plus sign (forward count)
PORDTMIN	X'40'	Minus sign (backward count)
PORDTABS	X'20'	No sign (start from the beginning)

Response-Record Return and Feedback Codes:

Return Code		Feedback Code		
Symbol	Value	Symbol	Value	Explanation
PORSROK	X'00'	PORSFOK	X'00'	Order accepted.
PORSRINV	X'08'			Order not accepted.
		PORSFPAR	X'01'	Parameter string is missing or invalid.
		PORSFONA	X'02'	Subsystem-internal reason.

Figure 56. Data Section and Return and Feedback Codes for a Restart-Device Order

Cancel-Output Order

VSE/POWER passes the order to your program when a PFLUSH DEV command is processed for a device under your program's control.

Figure 57 on page 189 shows the format of the device order's data section and the return and feedback codes that your program may have to supply in the response record.

Device-Order Data Section:

Bytes	Field Name	Contents / Description
20	PORDFFLG	HOLD indicator – HOLD was specified in the command if the byte is set to PORDFHLD (X'80'); else, the byte is set to X'00'.
21 - 22		Reserved.
23	PORDFPSL	Length of parameter string
24 - 5F	PORDFPRM	Parameter string

Response-Record Return and Feedback Codes:

Return Code		Feedback Code		
Symbol	Value	Symbol	Value	Explanation
PORSROK	X'00'	PORSFOK	X'00'	Order accepted.
PORSRINV	X'08'			Order not accepted.
		PORSFPAR	X'01'	Parameter string is missing or invalid.
		PORSFONA	X'02'	Subsystem-internal reason.

Figure 57. Data Section and Return and Feedback Codes for a Cancel-Output Order

Transmit-Command Order

VSE/POWER passes the order to your program when a PXMIT DEV command is processed for a device under your program's control. The command specified in the PXMIT command is passed to your program unchanged.

Figure 58 on page 190 shows the format of the device order's data section and the return and feedback codes that your program may have to supply in the response record.

Device-Order Data Section:

Bytes	Field Name	Contents / Description
20	PORDXPSL	Length of the specified command.
21 - A4	PORDXPRM	The command specified in the PXMIT command.

Response-Record Return and Feedback Codes:

Return Code		Feedback Code		
Symbol	Value	Symbol	Value	Explanation
PORSROK	X'00'	PORSFOK	X'00'	Order accepted.
PORSRINV	X'08'			Order not accepted.
		PORSFONA	X'02'	Subsystem-internal reason.

Figure 58. Data Section and Return and Feedback Codes for a Transmit-Command Order

Subsystem Orders

VSE/POWER accepts from your program and processes subsystem orders as follows:

- Send-message order
- Set-logical-destination order

To pass an order to VSE/POWER, your program must:

1. Set the buffer-type flag in the XPCCB to indicate that your program's send buffer contains a control record.

2. Ensure that the buffer contains the correct order-control record and nothing else.

3. Issue an XPCC request with FUNC=SENDR.

Your program can pass an order at any time after completion of a preceding request.

VSE/POWER replies to the order with the corresponding response record. Figure 59 on page 191 shows the return and feedback codes which VSE/POWER may set in its .response record.

Return Code		Feedback Code		
Symbol	Value	Symbol	Value	Explanation
PORSROK	X'00'	PORSFOK	X'00'	Order accepted.
PORSRINV	X'08'			Order not accepted.
		PORSFINV	X'01'	Order is invalid or unknown.
		PORSFOTS	X'02'	Order is too short.
		PORSFMSG	X'03'	Message text is too long.
		PORSFSLD	X'04'	Invalid destination in a preceding set-logical destination order.

Figure 59. Return and Feedback Codes for Subsystem Orders

Send-Message Order: Your program would pass a send-message order when it detects an error or an intervention-required condition on the involved device. This order includes the message that your program wants to be routed to the responsible operator or user.

VSE/POWER routes the message as instructed – to the system console if the order does not include a user identifier. It issues the message with all alphabetic characters converted to uppercase.

If the message cannot be forwarded to its final destination, then VSE/POWER discards the message without informing your program. Therefore, if your program requires a reply to the message, be sure to supply the identifier of a user that you know to be online.

The data section of a send-message order (labeled PORDMSG) contains the free-format message as set up by your program. This message can have a length of up to 120 alphameric characters.

Set-Logical-Destination Order: A user can route a job's output to a certain destination. This is done by specifying, in an * $$ LST (* $$ PUN) statement for the output, a user identifier with or without a node name. If this identifier is the name of a device under your program's control, then the output is selectable for processing by your program.

By way of a set-logical-destination order, you can instruct VSE/POWER to "equate" up to eight names to the one by which the involved output device is known in your program. VSE/POWER then selects an output for processing by this device if it is destined for an equated user.

However, if the original name by which the output device is known in your program is to be used as userid for routing output further, that name must be included in the list of logical destinations. An operator who issued a PSTART DEV command for a device can control that device only by commands using the same device name.

You may define identical logical destinations for several (or all) devices used under your program's control for the processing of spooled output. If you do this, two or more of these devices are available for the processing of output for certain logical destinations. In other words, you get a certain pool effect for your output devices. Consider this if you see a need for load levelling for the involved output devices.

You can pass a set-logical-destination order for a device at any time after this device has been started in response to a start-device order.

VSE/POWER uses the defined logical destination names when your program passes the next generic GET-open service request via the same communication path. Therefore, code a set-logical-destination order followed by a generic GET-open service request at the point where your program finds VSE/POWER's service task waiting for work. The set-logical-destination order may make one or more output queue entries selectable for processing by your program.

In a set-logical-destination order, bytes 0 through 3 of the header section are used as shown in Figure 50 on page 180; the remaining bytes of this section are of no significance. The order's data section, an area of 64 bytes at label PORDDLOG, is used for the definition of logical destinations, names of up to eight alphameric characters, as follows:

1. Fill the area with blanks.

2. Specify the destination names, one after the other and one per name slot of eight bytes. Include the logical name of the output device, if necessary. For VSE/POWER, a blank in the first character position of a name slot means that no more names follow.

Process a Signal

Signals supply status information required at the other end of a communication path. VSE/POWER and your program can work with status signals as follows:

- Output-arrived signal

  VSE/POWER passes this signal to your program when an output queue entry has become available for processing on the involved device.

If you operate in a shared-spooling environment, this output may have been placed into the output queue by one of the other sharing systems.

The format of this signal, a control record, is shown in Figure 60. VSE/POWER passes the record as the only one to your program's reply buffer for the communication path after a wait-for-order/signal or return-order/signal request. VSE/POWER needs no specific response after having passed an output-arrived signal.

Note: A generic GET-open service request in response to an output-arrived signal may nevertheless result in a "no entry available" response by VSE/POWER. Another user of your system may have requested that this selectable output queue entry be processed, or the entry's class may have changed.

- Device-stopped signal

  VSE/POWER expects this signal from your program after (but not necessarily in immediate response to) a stop-device order. Your program should pass the signal to VSE/POWER after all available records have been processed on the involved device.

Bytes	Field Name	Contents / Description
0 - 1	PSGNRLEN	Record length.
2	PSGNLTYP	X'07' - Signal-control record indicator.
3	PSGNLMOD	X'01' - Output-arrived indicator.
4 - 7		Reserved.

Figure 60. Output-Arrived Signal Control Record

- Setup-processed signal

  VSE/POWER expects this signal from your program after (but not necessarily in immediate response to) a setup-device order. Your program should pass the signal to VSE/POWER when the program's processing for the necessary setup activity is complete.

To pass a signal to VSE/POWER, your program must:

1. Set up a null send buffer.

2. Set byte PXUBTYP of the XPCCB to zero.

3. Set byte PXUSIGNL of the XPCCB to PXUSDSTP (for device-stopped) or PXUSSET (for setup processed).

4. Issue an XPCC request specifying FUNC=SENDR.

5. Check the return codes in register 15 and in the XPCCB byte IJBXRETC.

6. Issue a WAIT IJBSECB.

7. When the ECB is posted, VSE/POWER has returned a null buffer and passed return/feedback codes in the XPCCB user data. Check the VSE reason code in field IJBXREAS, and the VSE/POWER return and feedback codes.

GENERAL HINTS

The following remarks generally apply to using the external device support.

Routing of VSE/POWER-Generated Messages for External Devices

If the device owner issuing the PSTART DEV,devname command is not the local central operator but, for example, a remote-node operator (or an authorized subsystem administrator), then VSE/POWER routes all messages concerning the device status to

1. The device owner (PSTART DEV operator), **and** to

2. The central operator, if required by the severity of the message, or even to

3. The command originator, if DEV-type commands for an already started output device originated from a third party.

Range of Support for Communicating with a Subsystem

Throughout the preceding discussion of the external device support it was assumed that, to process an output queue entry, your program would normally issue a generic GET-service request with PWRSPL...QUEUE=LST specified. However, it is also possible to issue a GET request to the PUN queue, as well as a CTL request to any of the VSE/POWER queues. GET requests to the RDR queue and PUT requests are not allowed.

No password checking is done for a queue entry that is to be processed under subsystem control.

Use of VSE/POWER Commands During Program Debug Activities

As a help in program debugging, you can consult the output as displayed by the following commands:

- PDISPLAY A,DEV
- PINQUIRE DEV={devname|ALL}

For both commands, see the examples in VSE/POWER Installation and Operations Guide, following the description of the respective commands.

Use the PSTOP DEV,devname,FORCE command if you want to force an immediate termination of the communication path to a subsystem device.

APPENDIX A. READER EXIT ROUTINE

VSE/POWER supports user-written reader exit routines for local input and for input from the network. This appendix discusses the routine for local input; for a discussion of the exit routine for networking input, see the publication VSE/POWER Networking User's Guide.

To write a reader-exit routine for local input, consider the following:

- Conversion of control statements to uppercase characters

  VSE/POWER performs this conversion for all VSE/POWER JECL statements and for the VSE job control statements // JOB and // EXEC.

- Passing of control to the exit routine

  The routine receives control from VSE/POWER when a VSE job control or a VSE/POWER JECL statement is being read, either from a local or a remote reader, but not if DISP=I is used. VSE/POWER passes to the routine statements as listed below, including continuation lines, if any:

  ```
  All statements beginning with //
  All statements beginning with /.
  All statements beginning with *
  /*
  /&
  ```

- Coding conventions for the routine

  The routine must be re-enterable if two or more VSE/POWER reader tasks are running at the same time. A reader task in this context is:

  - A task started by a PSTART RDR,... command.
  - A task servicing a job-input spool request from another partition.
  - An RJE reader task started when a work station sends job data to VSE/POWER.

  The routine may change or delete any VSE job control or JECL statement; it may insert other statements. The routine may not perform an operation that causes a wait condition.

  If the routine inserts a statement, VSE/POWER handles this statement and then passes to the routine once more the original statement. After having made all the insertions, the routine must indicate whether to delete or process the original statement.

VSE/POWER processes each statement as it is received from the routine.

Note: If the routine inserts statements after it has passed the * $$ EOJ statement, VSE/POWER runs out of processor storage during end-of-job processing, and the operator gets a wait message. Be sure to avoid a situation such as this.

When the exit routine gets control from VSE/POWER, register 0 contains the address of the statement read and register 1 the length of the statement. To return to VSE/POWER, issue a BR 14 instruction.

The exit routine may not alter the contents of registers 10, 11, 12, and 13. These registers are reserved for VSE/POWER. Register 11 points to the task control block of the read task and may be used to identify the task.

The exit routine must return control to VSE/POWER with one of the following return codes in register 15:

X'00' Process the statement passed to the routine. The exit routine may update fields within the statement but may not change its length or address.

X'04' Ignore this statement.

X'08' Insert and process the new statement; return the original statement to the exit routine once more. Any number of statements may be inserted.

The address of the statement that is to be inserted must be provided in register 0, the statement's length in register 1. This length must be X'50'.

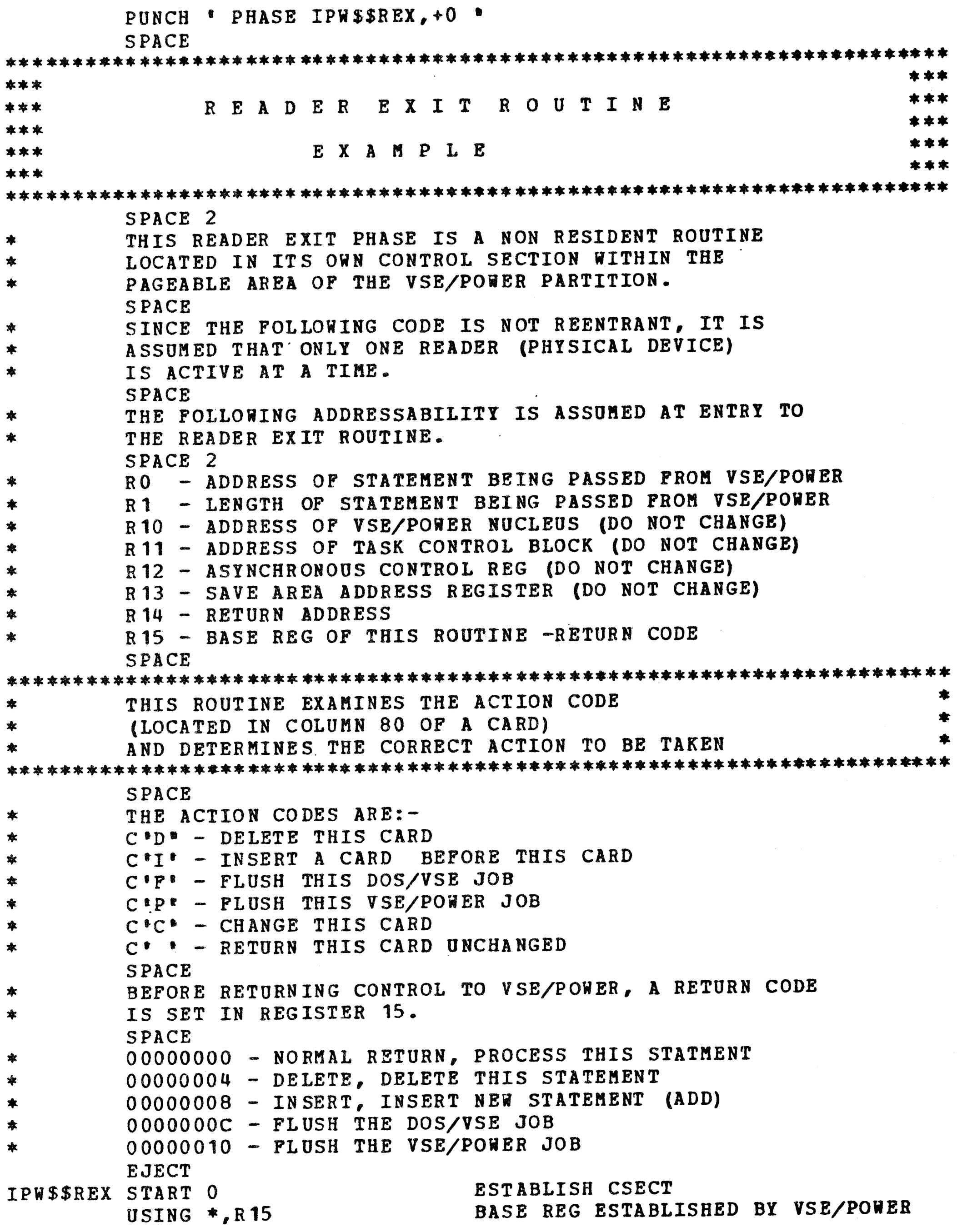
X'0C' Terminate the VSE job. This code is valid only when VSE/POWER passed a // JOB statement.

X'10' Terminate the VSE/POWER job.

Termination conditions (return codes X'0C' and X'10') at VSE/POWER job boundary (first statement of a VSE/POWER job) are ignored, and VSE/POWER issues a message.

If ACCOUNT=YES was specified, the number-of-records count in the reader account record reflects records added or deleted by the exit routine.

Figure 61 gives a coding example for a reader exit routine.

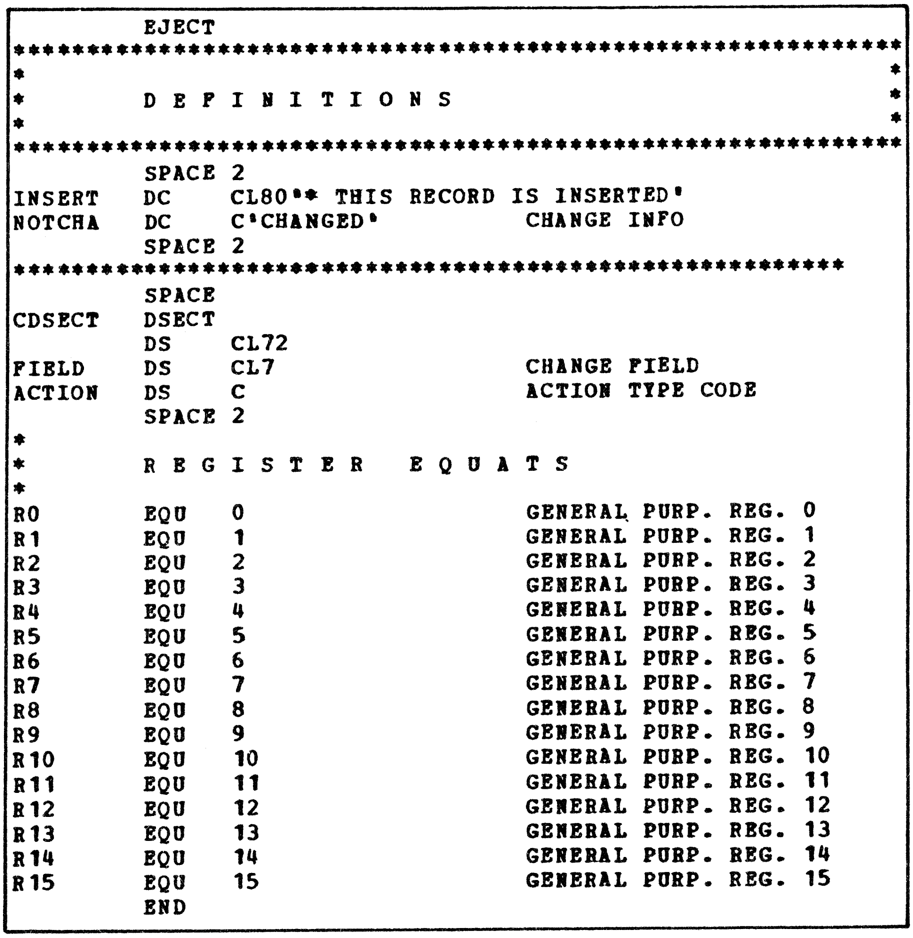


```
         PUNCH ' PHASE IPW$$REX,+0 '
         SPACE
***********************************************************************
***                                                                 ***
***           R E A D E R   E X I T   R O U T I N E                 ***
***                                                                 ***
***                   E X A M P L E                                 ***
***                                                                 ***
***********************************************************************
         SPACE 2
*        THIS READER EXIT PHASE IS A NON RESIDENT ROUTINE
*        LOCATED IN ITS OWN CONTROL SECTION WITHIN THE
*        PAGEABLE AREA OF THE VSE/POWER PARTITION.
         SPACE
*        SINCE THE FOLLOWING CODE IS NOT REENTRANT, IT IS
*        ASSUMED THAT ONLY ONE READER (PHYSICAL DEVICE)
*        IS ACTIVE AT A TIME.
         SPACE
*        THE FOLLOWING ADDRESSABILITY IS ASSUMED AT ENTRY TO
*        THE READER EXIT ROUTINE.
         SPACE 2
*        R0  - ADDRESS OF STATEMENT BEING PASSED FROM VSE/POWER
*        R1  - LENGTH OF STATEMENT BEING PASSED FROM VSE/POWER
*        R10 - ADDRESS OF VSE/POWER NUCLEUS (DO NOT CHANGE)
*        R11 - ADDRESS OF TASK CONTROL BLOCK (DO NOT CHANGE)
*        R12 - ASYNCHRONOUS CONTROL REG (DO NOT CHANGE)
*        R13 - SAVE AREA ADDRESS REGISTER (DO NOT CHANGE)
*        R14 - RETURN ADDRESS
*        R15 - BASE REG OF THIS ROUTINE -RETURN CODE
         SPACE
***********************************************************************
*        THIS ROUTINE EXAMINES THE ACTION CODE                        *
*        (LOCATED IN COLUMN 80 OF A CARD)                             *
*        AND DETERMINES THE CORRECT ACTION TO BE TAKEN                *
***********************************************************************
         SPACE
*        THE ACTION CODES ARE:-
*        C'D' - DELETE THIS CARD
*        C'I' - INSERT A CARD  BEFORE THIS CARD
*        C'F' - FLUSH THIS DOS/VSE JOB
*        C'P' - FLUSH THIS VSE/POWER JOB
*        C'C' - CHANGE THIS CARD
*        C' ' - RETURN THIS CARD UNCHANGED
         SPACE
*        BEFORE RETURNING CONTROL TO VSE/POWER, A RETURN CODE
*        IS SET IN REGISTER 15.
         SPACE
*        00000000 - NORMAL RETURN, PROCESS THIS STATMENT
*        00000004 - DELETE, DELETE THIS STATEMENT
*        00000008 - INSERT, INSERT NEW STATEMENT (ADD)
*        0000000C - FLUSH THE DOS/VSE JOB
*        00000010 - FLUSH THE VSE/POWER JOB
         EJECT
IPW$$REX START 0                   ESTABLISH CSECT
         USING *,R15               BASE REG ESTABLISHED BY VSE/POWER
```

Figure 61 (Part 1 of 3). Coding Example of a VSE/POWER Reader Exit Routine

```
         LR    R2,R0               GET ADDR OF RECORD
*
*
*        NOTE:
*              IT IS NOW THE USERS RESPONSIBILITY TO
*              DEVELOP HIS OWN ROUTINE CONCERNING HIS
*              PROBLEM DEFINITION.
*
*
         EJECT
***********************************************************************
*                                                                     *
*        THE FOLLOWING PIECE OF CODE IS USED TO OBTAIN THE            *
*        ACTION TYPE AND GETS THE RELATED RETURN CODE FOR             *
*        VSE/POWER                                                    *
*                                                                     *
***********************************************************************
         EJECT
IPWNED   DS    0H                  ENTRY FOR TEST FOR INSERT
         USING CDSECT,R2           ESTABLISH ADDRESSABILITY
         SPACE
         LA    R3,4                ASSUME DELETE
         CLI   ACTION,C'D'         DO WE WANT TO DELETE THIS CARD
         BE    IPWEXI              BRANCH IF YES
         SPACE
         CLI   ACTION,C'I'         DO WE WANT TO INSERT
         BNE   IPWFLT              ... BRANCH IF NOT
         LA    R3,8                SET PROPER RETURN CODE
         MVI   ACTION,C' '         BLANK OUT ACTION CODE
         LA    R0,INSERT           POINT TO CORRECT CARD
         LA    R1,L'INSERT         GET PROPER LENGTH
         B     IPWEXI              RETURN TO VSE/POWER
IPWFLT   DS    0H
         LA    R3,12               ASSUME FLUSH DOS/VSE JOB
         CLI   ACTION,C'F'         DO WE WANT TO FLUSH DOS/VSE JOB
         BE    IPWEXI              BRANCH IF YES
         LA    R3,16               ASSUME FLUSH OF VSE/POWER JOB
         CLI   ACTION,C'P'         DO WE WANT TO FLUSH VSE/POWER JOB
         BE    IPWEXI              BRANCH IF YES
         SPACE
         CLI   ACTION,C'C'         DO WE WANT TO CHANGE THIS CARD
         BNE   IPWNCH              ... BRANCH IF NOT
         MVI   ACTION,C' '         BLANK ACTION CODE
         MVC   FIELD,NOTCHA        MOVE IN CHANGE INFORMATION
IPWNCH   DS    0H
         SR    R3,R3               GET NORMAL RETURN CODE
*
*        RETURN TO VSE/POWER
*
IPWEXX   DS    0H
         LR    R15,R3              SET RETURN CODE
         BR    R14                 RETURN TO VSE/POWER
         DROP  R2                  RELEASE ADDRESSABILITY
```

Figure 61 (Part 2 of 3). Coding Example of a VSE/POWER Reader Exit Routine



```
         EJECT
*************************************************************
*                                                           *
*        D E F I N I T I O N S                              *
*                                                           *
*************************************************************
         SPACE 2
INSERT   DC    CL80'* THIS RECORD IS INSERTED'
NOTCHA   DC    C'CHANGED'          CHANGE INFO
         SPACE 2
*********************************************************
         SPACE
CDSECT   DSECT
         DS    CL72
FIELD    DS    CL7                 CHANGE FIELD
ACTION   DS    C                   ACTION TYPE CODE
         SPACE 2
*
*        R E G I S T E R   E Q U A T S
*
R0       EQU   0                   GENERAL PURP. REG. 0
R1       EQU   1                   GENERAL PURP. REG. 1
R2       EQU   2                   GENERAL PURP. REG. 2
R3       EQU   3                   GENERAL PURP. REG. 3
R4       EQU   4                   GENERAL PURP. REG. 4
R5       EQU   5                   GENERAL PURP. REG. 5
R6       EQU   6                   GENERAL PURP. REG. 6
R7       EQU   7                   GENERAL PURP. REG. 7
R8       EQU   8                   GENERAL PURP. REG. 8
R9       EQU   9                   GENERAL PURP. REG. 9
R10      EQU   10                  GENERAL PURP. REG. 10
R11      EQU   11                  GENERAL PURP. REG. 11
R12      EQU   12                  GENERAL PURP. REG. 12
R13      EQU   13                  GENERAL PURP. REG. 13
R14      EQU   14                  GENERAL PURP. REG. 14
R15      EQU   15                  GENERAL PURP. REG. 15
         END
```

Figure 61 (Part 3 of 3). Coding Example of a VSE/POWER Reader Exit Routine

APPENDIX B. VSE/POWER SPOOL-MACRO SUPPORT

This Appendix describes the XECB-macro based cross-partition communication (SPOOL-macro) support. This support has been available for accessing VSE/POWER services from within a program; it is retained to ensure program compatibility, and you can use it side by side with the newly available XPCC-macro based support described in Chapter 2. Continued use of the SPOOL-macro support requires that, for VSE/POWER table generation, you specify SPOOL=YES in the POWER macro.

For using the SPOOL-macro support, macros are available as listed below. For a description of these macros, refer to the indicated sections.

CTLSPOOL	See "CTLSPOOL: Control VSE/POWER Jobs" on page 209.
GETSPOOL	See "GETSPOOL: Retrieve Data from the Queues" on page 215.
PUTSPOOL	See "PUTSPOOL: Submitting a Job Stream" on page 220.
SPL	See "SPL: Generate a Spool Parameter List" on page 205.

In addition, you need the VSE macro XECBTAB, which is described in the publication *VSE/Advanced Functions, Application Programming: Macro Reference*.

To connect to VSE/POWER, use the VSE XECBTAB macro with the following operands:

```
XECBTAB TYPE=DEFINE,
        XECB={SPMXECB|ICRXECB}
        ACCESS=XWAIT
```

Specify XECB=SPMXECB for a GETSPOOL or a CTLSPOOL macro, specify XECB=ICRXECB for a PUTSPOOL macro. An XECB (cross-partition event control block) must be at least eight bytes long.

VSE/POWER requires the three-byte address of a spool parameter list (SPL) to be inserted into the XECB before your program issues a service request. You insert this address at:

SPMXECB+5 for a GETSPOOL or a CTLSPOOL request.
ICRXECB+5 for a PUTSPOOL request.

Other than XECBTAB, no VSE macro is required for VSE/POWER's SPOOL-macro support. Issue an XECBTAB=DELETE for the defined XECBs when the support is no longer required by your program.

Coding Practices: Only one user of the PUTSPOOL macro, and only one user of either the GETSPOOL or the CTLSPOOL macro may be active at any point in time. You can bypass this restriction and also avoid many a contention situation by using the support described in Chapter 2 on page 29.

SPOOL Macro Support

For the conventions used in presenting macro formats in this appendix, refer to Chapter 1 of VSE/POWER Installation and Operations Guide. A coding example for using the SPOOL-macro support is given under "Coding Example for Using the SPOOL-Macro Support" on page 226.

Notes:

1. VSE/POWER responds to spooling requests from the SVA. However, the required SPLs and data areas must reside in the partitions that contain the requesting programs.

2. A program using the SPOOL-macro support must include an SPL TYPE=MAP macro.

3. The operand PBUF=buffaddr must be specified in either the definition macro SPL, or in the execution macro (CTLSPOOL, GETSPOOL, or PUTSPOOL) that is executed first in the program.

4. A system error may occur if:

   a. The partition using the SPOOL-macro support has a higher priority than VSE/POWER.
   b. An abnormal end or shut down of VSE/POWER occurs before all active SPOOL-macro service tasks have completed.

5. Before using the support, you must save your registers 0, 1, 13, 14, and 15 (they are used and overwritten by VSE/POWER). Register 15 contains the return code.

6. The operands CLASS= and DISP= are not supported in the PUTSPOOL macro or its associated SPL.

   To specify the output class or disposition for a job submitted by PUTSPOOL, include an * $$ LST or * $$ PUN statement at the beginning of the job. If you supply an * $$ JOB statement, include the * $$ LST or * $$ PUN statement immediately behind the * $$ JOB statement.

   If these PUTSPOOL operands have already been coded in an existing program, VSE/POWER updates the SPL with the specified value (in case the SPL is used later by a GETSPOOL or CTLSPOOL macro).

   If you use these operands in a modified source program, then:

   - You receive a warning comment if they occur in the SPL macro.
   - You receive an assembler generated MNOTE if they occur in the PUTSPOOL macro.

SPL: Generate a Spool Parameter List

The macro builds a spool parameter list (SPL) for use by the execution macros PUTSPOOL, GETSPOOL, and CTLSPOOL. Any specification you make in an SPL is in effect for the execution macro using this SPL, except if (a) the specification is overridden by a corresponding operand of the execution macro or (b) the REQ= operand is not specified in the CTLSPOOL macro; in that case the class specification is modified.

Correct use of the macro requires you to:

1. Store the SPL address into the correct XECB.

2. Load the pointer register named in the SPL=(reg) operand of the CTLSPOOL, GETSPOOL, or PUTSPOOL macro.

Format of the Macro

The macro can be used with two distinct sets of operands as shown below.

Name	Operation	Operands
[name]	SPL	TYPE=DEFINE [,CBUF=firstbuffaddr] [,CLASS={A|class}] [,DISP={K|disposition}] [,JOBN={DUMMY|jobname}] [,NEWVAL=value] [,PBUF=buffaddr] [,PBUFL={88|bufflength}] [,PWD=password] [,REQ={CANCEL| CLASS| COMMAND| DISP| LOOKUP| PRI| REMOTE| SCRATCH| STATUS|}] [,USERID=userid]
		TYPE=MAP [,ICRXECB={YES|NO}] [,SPMXECB={YES|NO}]

Description of Operands

TYPE=DEFINE

The operand causes an SPL to be set up with the specified values.

CBUF=firstbuffaddr

The operand specifies the address of the first buffer of a chain of buffers that contain the job stream. Each of the buffers has a length of 88 bytes and a format as follows:

Bytes 0 - 3 = Pointer to the next buffer in the chain; set to zero in the last buffer.
Bytes 4 - 7 = Reserved.
Bytes 8 - 87 = Spool record.

Up to 4095 such buffers may be chained for each PUTSPOOL access.

CLASS={A|class}

The operand specifies the VSE/POWER output class (A-Z) for the affected job. The operand is ignored if specified in an SPL for PUTSPOOL.

DISP={K|disposition}

The operand specifies the output disposition for the affected job. The operand is ignored if specified in an SPL for PUTSPOOL.

JOBN={DUMMY|jobname}

The operand specifies the job name to be assigned to the affected input queue entry for a PUTSPOOL operation or to be searched for in case of a CTLSPOOL or GETSPOOL operation.

NEWVAL=value

The operand is meaningful only if the SPL is to be used with CTLSPOOL.

For value in the operand, specify the new value that is to be assigned in accordance with your specification in the REQ operand of the CTLSPOOL macro. You can specify a new value for one of the following:

Class of the job:	REQ=CLASS in CTLSPOOL.
Disposition of the job:	REQ=DISP in CTLSPOOL.
Priority of the job:	REQ=PRI in CTLSPOOL.
A remote identifier:	REQ=REMOTE in CTLSPOOL.

The value can be specified in one of the following ways:

C'x' For example, NEWVAL=C'A'
X'nn' For example, NEWVAL=X'01'

PBUF=buffaddr

For buffaddr, specify the address of an area for use by VSE/POWER and for VSE/POWER feedback information on certain error conditions.

PBUFL={88|bufflength}

For bufflength, specify (in number of bytes) the length of the buffer whose address is given in PBUF=buffaddr. Define your buffer's length large enough for your longest data record to fit into the buffer. VSE/POWER truncates the trailing blanks of a record; it indicates the length of each record after truncation in either:

- The four-byte SPL field SPRL if data records are not blocked, or
- Bytes 2 and 3 of the record prefix if the data records are blocked (see also the MODE=BUF operand of the GETSPOOL macro).

The minimum length you can specify is 88.

PWD=password

The operand specifies the password that must be associated with the request.

If you omit this operand, VSE/POWER sets the SPL's password field to blanks. Should a password be required nevertheless, then supply this in your request macro (CTLSPOOL, GETSPOOL, or PUTSPOOL).

Note: The default password setting for access requests from a program is different from the default password setting for locally submitted jobs. Your program can access a locally submitted job (or its output) only if a password was assigned to this job explicitly and if your program presents this password in the request SPL.

The password can be any alphameric string of up to eight characters.

REQ={CANCEL|CLASS|COMMAND|DISP|LOOKUP|PRI|REMOTE|SCRATCH|STATUS}

The operand defines a default for CTLSPOOL requests. For a description of the various specifications, refer to the description of the CTLSPOOL macro.

USERID=userid

For userid, specify the identifier of the user who originates the affected queue entry, if this is possible. Normally, you would supply this identifier in the applicable PUTSPOOL request.

Supply a user identifier, an alphameric string of up to eight characters, whenever you submit a job to VSE/POWER. This helps preventing unauthorized access to the job or to output produced by the job.

TYPE=MAP

The operand causes a DSECT of the SPL to be generated. An SPL macro with TYPE=MAP must be specified at least once in a program using the SPOOL-macro support.

ICRXECB={YES|NO}

Specify ICRXECB=YES if the DSECT to be generated is to apply to an SPL for use with PUTSPOOL.

SPMXECB={YES|NO}

Specify SPMXECB=YES if the DSECT to be generated is to apply to an SPL for use with CTLSPOOL or GETSPOOL.

CTLSPOOL: Control VSE/POWER Jobs

The macro requests VSE/POWER to do one of the following:

- Alter the attributes of a VSE/POWER job.
- Cancel a submitted job prior to its execution.
- Delete the list or punch output of a job after its execution.
- Display the status of any job or of all jobs.
- Send a message to another user, remote operator, or central operator.
- Submit a VSE/POWER command for execution.

Nearly all of the macro's operands allow you to use register notation (indicated by "(reg)" as a possible specification). You can use for this purpose any register, except the registers 0, 1, 14, and 15.

Format of the Macro

Name	Operation	Operands
[name]	CTLSPOOL	SPL=(reg) [,CCLASS={value|(reg)}] [,JNUM={SPL|(reg)}] [,JOBN={jobname|(reg)}] [,MODE=SPOOL] [,NEWVAL={value|(reg)}] [,PBUF={buffaddr|(reg)}] [,PWD={password|(reg)}] [,QUEUE={LST|PUN|RDR|XMT}] [,REQ={CANCEL| CLASS| COMMAND| DISP| LOOKUP| PRI| REMOTE| SCRATCH| STATUS| (reg)}] [,USERID={userid|(reg)}]

Description of Operands

SPL=(reg)

This mandatory operand specifies the register which contains the address of the spool parameter list (SPL) to be used. The SPL defines the request to VSE/POWER.

CCLASS={value|(reg)}

For value, specify the class of the queue entries to which the CTLSPOOL request is to apply. You can specify the value in one of the following forms:

```
C'x'  For example, CCLASS=C'A'
X'nn' For example, CCLASS=X'F1'
```

If you use register notation, the specified register must contain the class in its low-order byte.

This operand is valid only with one of the following:

```
REQ=CLASS
REQ=DISP
REQ=PRI
REQ=REMOTE
```

JNUM={SPL|(reg)}

The operand specifies the job number that is to be used as a search argument together with the job name.

Specify JNUM=SPL if VSE/POWER is to use the job number currently stored in the SPL.

If you use register notation, the specified register must contain the job number.

If you omit this operand, VSE/POWER takes the first job with a matching name.

JOBN={jobname|(reg)}

For jobname, specify the name by which the affected job is known to VSE/POWER.

If you use register notation, the specified register must contain a pointer to an eight-byte storage field containing the job's name.

MODE=SPOOL

The operand causes VSE/POWER to write its response to the CTLSPOOL request into the LST queue and to return the LST queue entry's job name and number in the SPL used for the request. Issue a GETSPOOL request to retrieve this response from the LST queue.

MODE=SPOOL is valid only if REQ=COMMAND is specified and the submitted command is "PDISPLAY queue" or "PDISPLAY PNET." The job name assigned to the queue entry by VSE/POWER is $SPLnnnn (where nnnn = the job number assigned by VSE/POWER). The queue entry's class and disposition are the ones contained in the SPL.

NEWVAL={value|(reg)}

For value, specify the new value that is to be used by VSE/POWER as the job attribute. You can specify this value in one of the following forms:

C'x'	For example: NEWVAL=C'A'
X'nn'	For example: NEWVAL=X'F1'
n	For example: NEWVAL=5

The operand is valid only together with one of the following specifications:

REQ=CLASS	For a new class of the job.
REQ=DISP	For a new disposition of the job.
REQ=PRI	For a new priority of the job.
REQ=REMOTE	For a new remote identifier.

If you use register notation, the specified register must contain the new value.

PBUF={buffaddr|(reg)}

The operand specifies the address of a buffer which is for use by VSE/POWER and for VSE/POWER feedback information. The length of this buffer must be 88 bytes.

If you use register notation, the specified register must contain the buffer's address.

PWD={password|(reg)}

For password, specify the password associated with the affected VSE/POWER job or output.

If a password was defined on input or in an * $$ LST or * $$ PUN statement, then the same password is to be specified to have VSE/POWER execute any queue manipulation commands (such as PALTER or PDELETE). If there is no match of the passwords, then VSE/POWER rejects the request with a return code in the error/feedback bytes of the SPL for the request.

Omission of this operand (and also in the SPL macro) does not give you access to a queue entry submitted without password protection via a local spool device.

If you use register notation, the specified register must point to an eight-byte field that contains the password left-justified.

QUEUE={LST|PUN|RDR|XMT}

The operand specifies the queue to be used for the CTLSPOOL request:

LST	For list queue
PUN	For punch queue
RDR	For reader queue
XMT	For transmission queue

The operand is ignored if one of the following is specified

REQ=CANCEL
REQ=COMMAND
REQ=STATUS

REQ={CANCEL|CLASS|COMMAND|DISP|LOOKUP|PRI|REMOTE|SCRATCH|STATUS|(reg)}

The operand specifies the requested operation as follows:

CANCEL

Applies only to job input; causes the affected job to be deleted from the input queue if it has not yet been processed.

CLASS

Alters the job class of the job on the specified VSE/POWER queue. Requires a NEWVAL=value specification in order to be valid.

COMMAND

Indicates that you have supplied a VSE/POWER command in the area defined in the PBUF operand. No error detection is performed for the command, and no error code is returned, except for an invalid request (an invalid SPL address, for example). You must analyze the PBUF area in your program for a possible return message.

Your program can pass only one of the following commands per CTLSPOOL request:

PALTER queue,jobname	See "Note" below
PBRDCST	
PCANCEL jobname	See "Note" below
PDELETE queue,jobname	See "Note" below
PDELETE MSG	
PDISPLAY queue,jobname	
PDISPLAY MSG	
PDISPLAY A	
PDISPLAY T	
PDISPLAY PNET	
PHOLD queue,jobname	See "Note" below
PINQUIRE	
PRELEASE queue,jobname	See "Note" below
PXMIT	

Note: The command can be used in a networking environment for execution at another node if that other node is controlled by VSE/POWER.

VSE/POWER processes the command on your own VSE node if:

- Both the user-ID and the password match the user-ID and password specified for the job or its output, or
- At least the supplied password matches the password defined for the job or its output, should the user-ID not match.

On another node controlled by VSE/POWER, the command is presented only if the user-ID matches the one specified for the affected job or its output.

DISP
Alters the disposition of the affected queue entry. Requires a NEWVAL=value specification in order to be valid.

LOOKUP
Causes status information about the specified job or output to be returned in applicable fields of the SPL. VSE/POWER returns the following information:

Job number
Class
Disposition
Number of lines or cards
Flag (indicating that more than one queue entry exists)

PRI Alters the priority of the affected queue entry. Requires a NEWVAL=value specification in order to be valid.

REMOTE
Alters the remote identifier to which output of the job is to be routed. Requires a NEWVAL=value specification in order to be valid.

SCRATCH
Causes the named job to be deleted from the affected VSE/POWER output (LST, PUN, or XMT) queue.

STATUS
Causes the following to be passed to the named SPL:

- The disposition of the named job in the field SPQD of the SPL.
- The job's queue indicator in the field SPSQ of the SPL. This indicator may be:

  L = The job is in the LST queue.

N = Nothing to display (the specified job name is unknown).
P = The job is in the PUN queue.
R = The job is in the RDR queue.
X = The job is in the XMT queue.

If the job exists in more than one queue, only its first occurrence is returned. The queues are searched in this sequence: LST, RDR, PUN, XMT.

(reg)
Indicates that a request code is provided in the specified register. You can specify one of the following codes in this register:

Code	Request Type	Requested Function	Corresponding Command
X'01'	PRI	Alter the priority	PALTER
X'02'	DISP	Alter the disposition	PALTER
X'04'	CLASS	Alter the class	PALTER
X'08'	REMOTE	Alter the remote identifier	PALTER
X'10'	CANCEL	Cancel input	PDELETE RDR
X'20'	SCRATCH	Scratch output	PDELETE queue
X'40'	STATUS	Display the status of the named job	PDISPLAY
X'80'	COMMAND	Process the passed VSE/POWER command	–

USERID={userid|(reg)}
For userid, specify the user identifier of the queue entry that is to be manipulated. This identifier was defined when the job was submitted to VSE/POWER. If you omit the operand, VSE/POWER uses, for your CTLSPOOL request, the user identifier currently stored in the request SPL.

VSE/POWER rejects your request if:

- You specified an identifier which does not match the originally defined one.
- You did not specify an identifier, but the identifier currently stored in the SPL does not match the originally defined one.
- You did not specify an identifier, no identifier is stored in the SPL, and an identifier was defined for the affected queue entry.

If you use register notation, the specified register must point to an eight-byte field that contains the identifier left-justified.

GETSPOOL: Retrieve Data from the Queues

The macro requests the retrieval of data currently held in VSE/POWER queues on disk. VSE/POWER returns the requested data to the buffer area of the partition issuing the GETSPOOL macro.

VSE/POWER accepts the request only if the affected queue entry's disposition is D or K. As for an output task, the entry's disposition is changed to L after processing if this disposition was K, the entry is deleted if this disposition was D. Therefore, before you can retrieve a queue entry processed by VSE/POWER previously, you must issue a CTLSPOOL request that changes this entry's disposition from L to K again.

If you use GETSPOOL and do not read to the end of data, a problem can occur. The accessed queue entry remains in the VSE/POWER queue in an active state and the operator is unable to delete the entry (VSE/POWER displays DISP=*). You can avoid this by one of the following actions:

- Delete the entry using a CTLSPOOL request.
- Submit a CTLSPOOL request following your GETSPOOL, for example:

  ```
  CTLSPOOL SPL=(reg),REQ=STATUS
  ```

- Always read a queue entry until end-of-data.
- Request a GETSPOOL operation for another queue entry.

Any of these actions causes the entry to be deleted (disposition was D) or closed and retained with disposition L (disposition was K).

In response to the first GETSPOOL request, VSE/POWER returns, in your SPL, the number of records which the entry contains.

When end of data is reached, VSE/POWER returns the EOF indicator as a dummy record after the last data record. With buffered GETSPOOL requests, VSE/POWER returns the EOF indicator in the prefix of the last data record.

Format of the Macro

Name	Operation	Operands
[name]	GETSPOOL	SPL=(reg)
		[,CC={YES|NO}]
		[,CLASS={class|(reg)}]
		[,JNUM={SPL|(reg)}]
		[,JOBN={jobname|(reg)]
		[,LINENO={number|(reg)}]
		[,MODE=BUF]
		[,PBUF={buffaddr|(reg)}]
		[,PBUFL={bufflength|(reg)}]
		[,PWD={password|(reg)}]
		[,QUEUE={LST|PUN}]
		[,USERID={userid|(reg)}]

Description of Operands

SPL=(reg)

This mandatory operand specifies the register which contains the address of the SPL to be used. The SPL defines the request to VSE/POWER.

If you used the LINENO operand in a preceding GETSPOOL request, specify the address of the same SPL in this request; else, line positioning gets lost.

CC={YES|NO}

Specify CC=YES to have VSE/POWER return the command code of the CCW for the currently processed data record. VSE/POWER inserts this code in the field SPCC of the SPL, except when you specify also MODE=BUF.

If you specify also MODE=BUF, VSE/POWER passes all command codes, including those which have no associated data, to your program's buffer.

CLASS={class|(reg)}

For class, specify the class value assigned to the queue entry (by PUTSPOOL, for example). If you use register notation, supply the applicable class value in the specified register.

JNUM={SPL|(reg)}

Use this operand if two or more jobs in the accessed queue have the same name.

Specify JNUM=SPL if VSE/POWER is to use the job number currently stored in the SPL. Supply the VSE/POWER-assigned job number in the specified register otherwise. If you omit the operand, VSE/POWER sets the SPL's job number field to zero.

JOBN={jobname|(reg)

For jobname, specify the name by which VSE/POWER knows the queue entry that is to be retrieved. It is the name that was assigned to the entry (by PUTSPOOL, for example).

If you use register notation, the specified register must point to an eight-byte field containing the name in that field left-adjusted.

LINENO={number|(reg)}

Use this operand in your first GETSPOOL request for a queue entry if retrieval is to begin with a certain output record. The operand causes retrieval to begin at the specified line number relative to the beginning of the file. If you use register notation, supply the line number in the specified register.

The maximum value that you can specify for number is 16777215.

If you omit the operand, VSE/POWER starts retrieval at the beginning of the queue entry's spool data.

Do not use this operand in a second or subsequent GETSPOOL request for the same queue entry. Specifying the operand in a subsequent GETSPOOL request causes VSE/POWER's line pointer to be repositioned. Ensure, however, that the subsequent GETSPOOL requests use the same SPL as the initial request for this retrieval operation.

MODE=BUF

Specify this operand if VSE/POWER is to retrieve more than one record per request. The operand causes VSE/POWER to fill the area named in the PBUF operand with as many records as will fit. Each data record in that area has a four-byte prefix as follows:

Byte	Contents
0	Command code. If VSE/POWER is to pass also command-code-only records (such as a skip to channel 1), you must specify CC=YES in addition.
1	X'80' = The last record in the buffer. X'C0' = The last record of the spool data.
2-3	Length (in binary) of a data record, including the four-byte prefix.

Deblocking is to be done in your program.

PBUF={buffaddr|(reg)}

For buffaddr, specify the symbolic address of the buffer into which VSE/POWER is to pass retrieved data records or feedback information (on certain error conditions) or both. If you use this operand, you must also specify PBUFL=bufflength.

You can omit this operand and the PBUFL operand if you defined a buffer in your SPL.

If you use register notation, the specified register must point to the buffer which VSE/POWER is to use.

PBUFL={bufflength|(reg)}

The operand specifies the length (in number of bytes) of the buffer whose address is given in the PBUF operand. Define your buffer's length large enough for your longest data record to fit into the buffer. VSE/POWER truncates the trailing blanks of a record; it indicates the length of each record after truncation in either:

- The four-byte SPL field SPRL if data records are not blocked, or
- Bytes 2 and 3 of the record prefix if the data records are blocked (see also the MODE=BUF operand).

The minimum length you can specify is 88.

If you use register notation, the specified register must contain the buffer's length.

PWD={password|(reg)}

For password, specify the password associated with the queue entry that is to be retrieved. If there is no match of the passwords, then VSE/POWER rejects the request with a return code in the error/feedback bytes of the SPL for the request.

Omission of this operand (and also in the SPL macro) does not give you access to a queue entry submitted without password protection via a local spool device.

If you use register notation, the specified register must point to an eight-byte field that contains the password left-justified.

QUEUE={LST|PUN}

The operand specifies the queue to which the GETSPOOL request applies:

LST	For list queue
PUN	For punch queue

USERID={userid|(reg)}

For userid, specify the user identifier associated with the queue entry that is to be retrieved. This identifier was defined when the job was submitted to VSE/POWER. If you omit the operand, VSE/POWER uses, for your GETSPOOL request, the identifier currently stored in the request SPL.

If you use register notation, the specified register must point to an eight-byte field that contains the identifier left-justified.

PUTSPOOL: Submitting a Job Stream

You use the macro to submit a job stream from your program's buffer to the:

- VSE/POWER input (RDR) queue for later execution of this job stream in a partition under control of VSE/POWER

- VSE/POWER transmission (XMT) queue for transmission of this job stream to another node.

VSE/POWER analyses only those JECL statements which you submit with the first PUTSPOOL request for a queue entry. VSE/POWER places these statements into the input queue. For example, if you wish to specify output characteristics (such as class or disposition) other than the default values, supply an * $$ LST or * $$ PUN statement.

If your program does not pass an * $$ JOB statement, VSE/POWER builds this statement (in accordance with your specifications for the applicable SPL) and inserts it into your job stream.

For the second and subsequent PUTSPOOL requests, VSE/POWER passes your input from the buffer to the VSE/POWER input queue. No more checking is performed. When the last statement of the input has been read from the buffer and no more continuation input exists, VSE/POWER inserts an * $$ EOJ statement if one has not been passed.

The job number assigned by VSE/POWER is returned to your program in the job-number field of the SPL. You may want to use this job number later together with the job name in order to retrieve the job's output.

An * $$ RDR statement may be inserted into the data submitted via PUTSPOOL. Thus, data from a diskette can be included into the input job stream. However, PUTSPOOL communication is blocked for the duration of this diskette input. It is also blocked if the diskette unit is not ready or a wrong diskette is inserted.

If there is a user-written RDREXIT routine for local input, VSE/POWER passes to this routine the VSE job-control statements and JECL statements of the submitted jobs.

Format of the Macro

Name	Operation	Operands
[name]	PUTSPOOL	SPL=(reg) [,CBUF={firstbuffaddr|(reg)}] [,CONT=(reg)] [,JOBN={jobname|(reg)] [,PBUF={buffaddr|(reg)}] [,PWD={password|(reg)}] [,USERID={userid|(reg)}]

Description of Operands

SPL=(reg)
For reg, specify the register that contains the address of the SPL that is to be used by the PUTSPOOL macro. The SPL defines the request to VSE/POWER.

CBUF={firstbuffaddr|(reg)}
For firstbuffaddr, specify the symbolic address of the first of the 88-byte buffers that contain the job stream. The format of an 88-byte buffer is as follows:

Bytes	Contents
0-3	A pointer to the next buffer in the chain (0 for the last buffer).
4-7	Reserved.
8-87	An 80-byte data buffer area.

If the CONT operand is specified together with CBUF, the same buffers must be reused for the continuation data. If register notation is used, the continuation routine must reset the previously used register contents each time continuation data is made available. See also the description of the CONT operand below.

The CBUF pointer in the SPCB field of the SPL is used as a work area and is set to zero when PUTSPOOL processing is ended. If the PUTSPOOL macro uses an SPL which has already been used, or if the PUTSPOOL macro is entered more than once with the same SPL, then one of the following is required:

- CBUF is specified to reset the field SPCB of the SPL.

- The field SPCB of the SPL is updated (a maximum of 4,095 input buffers is allowed for each PUTSPOOL access).

- The buffer defined by CBUF is the only one (its chain pointer is zero).

CONT=(reg)
If the buffers processed by this execution of PUTSPOOL do not contain the complete job stream, this operand should be used to give the address of a continuation routine. In this routine, you can submit further data buffers associated with the same job stream. However, no other operands can be changed in the continuation routine.

When this operand is used, then also the CBUF operand must be specified.

To exit from the routine, set the specified register to zero and return to the PUTSPOOL macro or, via register 14, to VSE/POWER.

JOBN={jobname|(reg)
For jobname, specify the (unique) name that is to be assigned to the job in the VSE/POWER input queue. This name is to be used if, for example, the job's output is to be retrieved by a GETSPOOL macro or if the job is to be accessed by a CTLSPOOL macro.

PBUF={buffaddr|(reg)
For buffaddr, specify the address of a buffer for use by VSE/POWER and for VSE/POWER feedback information on certain error conditions. The size of this buffer must be at least 88 bytes. If register notation is used, the specified register must contain a pointer to this buffer.

PWD={password|(reg)}
Use this operand to define the password for this VSE/POWER job.

A password which you specify in a PUTSPOOL macro:

- Overrides the password that may be stored in the request SPL.
- Must be used in any subsequent GETSPOOL macro for the job.
- Can be overridden for output by the PWD operand of an * $$ LST statement or * $$ PUN statement.

The password can be any string of up to eight alphameric characters.

USERID={userid|(reg)}
For userid, specify the identifier which is to be associated with the queue entry that is to be placed into one of the VSE/POWER queues (RDR or XMIT).

If you use register notation, the specified register must point to an eight-byte field containing the identifier left-justified.

If you omit this operand and do not supply a user-ID with the SPL macro defining the request SPL, VSE/POWER spools the applicable job input without a user-ID.

Return Codes

Return Codes in the SPL: Figure 62 shows the return codes that your program receives following the execution of a PUTSPOOL, GETSPOOL, or CTLSPOOL macro. VSE/POWER supplies these codes as follows:

- In the SPL bytes SPER and SPER2.
- In a byte which you can access using the DSECT generated by SPL TYPE=MAP,...

  You access this byte by referring to field xxxXECB+4, where xxx is either SPM or ICR depending on the type of SPL.

Additional information is passed to your program in error-feedback byte 2 (SPER2) of the SPL:

Mnemonic of Equate	Hex. Value	Meaning
SPAI	80	Wrong password - access denied.
SPDDR	02	3540 data-mode record is being processed.

Return Codes in Register 15: VSE/POWER's SPOOL-macro support makes use of the VSE macros XPOST and XWAIT; your program should examine their return codes in register 15. For a description of these macros and their return codes, see the publication VSE/Advanced Functions, Application Programming: Macro Reference. XPOST return codes are multiplied by 16 to maintain code uniqueness.

Register 15 contains the code X'40' if VSE/POWER control tables were generated without SPOOL=YES specified in the POWER generation macro.

Return Code	Meaning	Passed in	
		xxxXECB +4	SPER
X'0x'	Miscellaneous:		
X'08'	End of data encountered during a GETSPOOL request, or invalid LINENO specified in GETSPOOL.	X	X
X'09'	Task was waiting on queue/account file space.		
X'1x'	Invalid specification:		
X'11'	Command not allowed.	X	X
X'12'	Invalid VSE/POWER output disposition in SPL.	X	X
X'14'	Invalid output class (not A-Z) in SPL.	X	X
X'16'	Invalid queue specified.	X	X
X'17'	Invalid password specified.	X	X
X'18'	Invalid job name in SPL.	X	X
X'2x'	Job processing errors:		
X'21'	The PBUF buffer area is smaller than 88 bytes or not large enough to hold the largest output data record (PBUFL in GETSPOOL too small).	X	X
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. If an invalid password was specified, X'80' appears in field SPER2 of the SPL also.	X	X
X'24'	A loop occurred in the PUTSPOOL buffer chain, or more than 4095 buffers were used per request.	X	X
X'28'	Invalid CTLSPOOL REQ operand.	X	X

Figure 62 (Part 1 of 2). Return Codes from CTLSPOOL, GETSPOOL, and PUTSPOOL

Return Code	Meaning	Passed in	
		xxxXECB +4	SPER
X'4x'	VSE/POWER diagnostic:		
X'41'	VSE/POWER terminated normally, or	X	
	VSE/POWER terminated abnormally, or	X	
	VSE/POWER spool management task terminated abnormally.	X	
X'42'	A VSE/POWER message was logged during CTLSPOOL (see notes below).	X	X
X'44'	A VSE/POWER error occurred during GETSPOOL (see notes below).	X	X
X'48'	A VSE/POWER error occurred during PUTSPOOL (see notes below).	X	X
X'49'	Task waiting on queue/account file space. This return code will not appear when the PUTSPOOL/ GETSPOOL user receives control back from VSE/POWER.		
X'8x'	Invalid address pointer:		
X'82'	Invalid data buffer chain (PUTSPOOL).	X	X
X'84'	Invalid VSE/POWER buffer address (PBUF).	X	X
X'88'	Invalid SPL address.	X	

1. The first 60 characters of the VSE/POWER message are displayed at displacement 28 of the buffer defined by the PBUF operand.

2. No spool management error detection is done for a CTLSPOOL with REQ=COMMAND. Your program must analyse the message returned by VSE/POWER in the buffer defined by the PBUF operand. If the command passed by a CTLSPOOL request results in more than one message, VSE/POWER returns only the message that best describes the condition.

3. All values specified in the NEWVAL operand of CTLSPOOL must conform to the related VSE/POWER rules.

Figure 62 (Part 2 of 2). Return Codes from CTLSPOOL, GETSPOOL, and PUTSPOOL

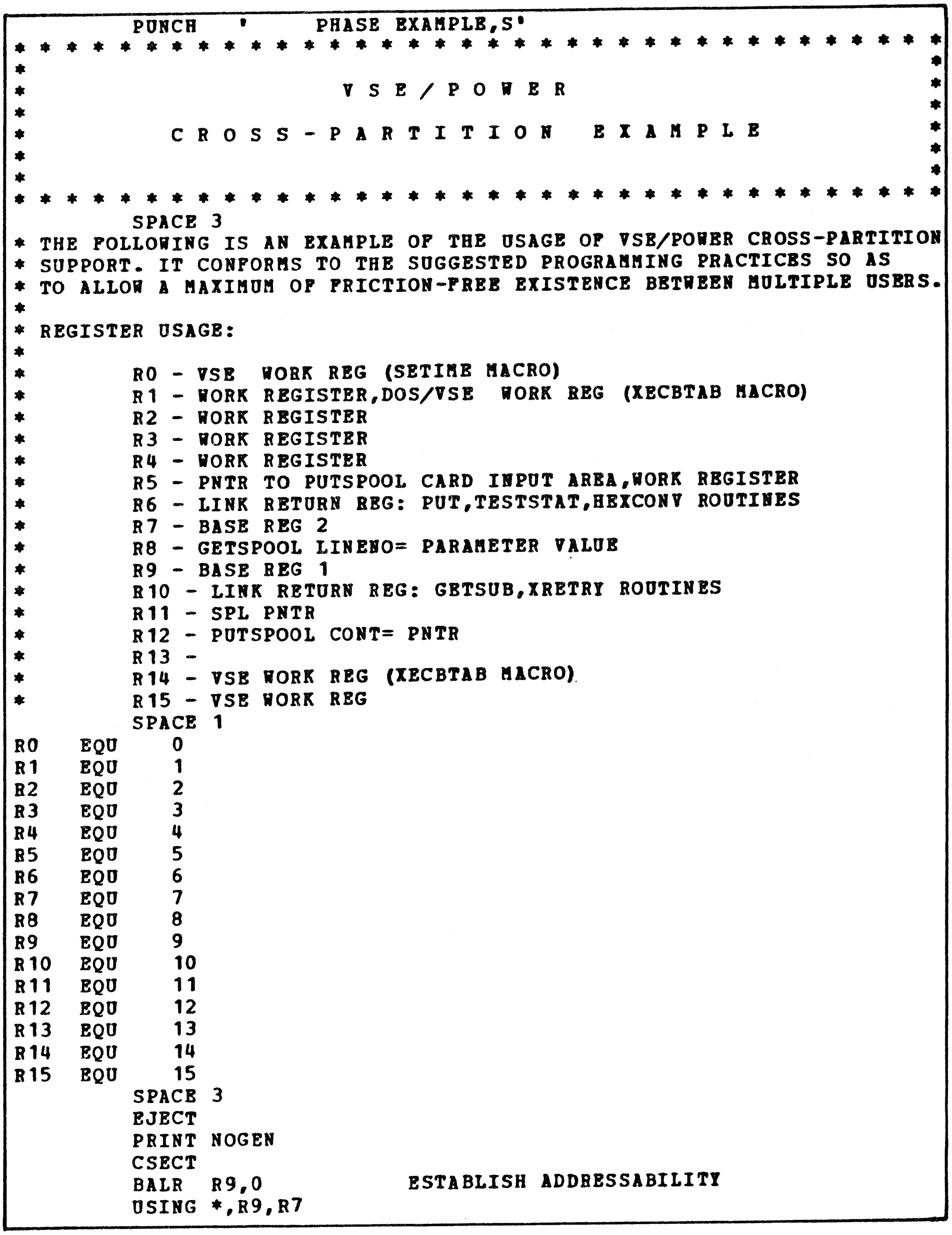
Coding Example for Using the SPOOL-Macro Support

Figure 63 gives a coding example for the use of the SPOOL-macro support. Figure 64 shows the console listing that resulted from running the example, and Figure 65 shows the corresponding list output.

The example submits a job made up of numbered card-image records. The output of the job is retrieved, with GETSPOOL, both sequentially and randomly. The output is displayed at the same time.

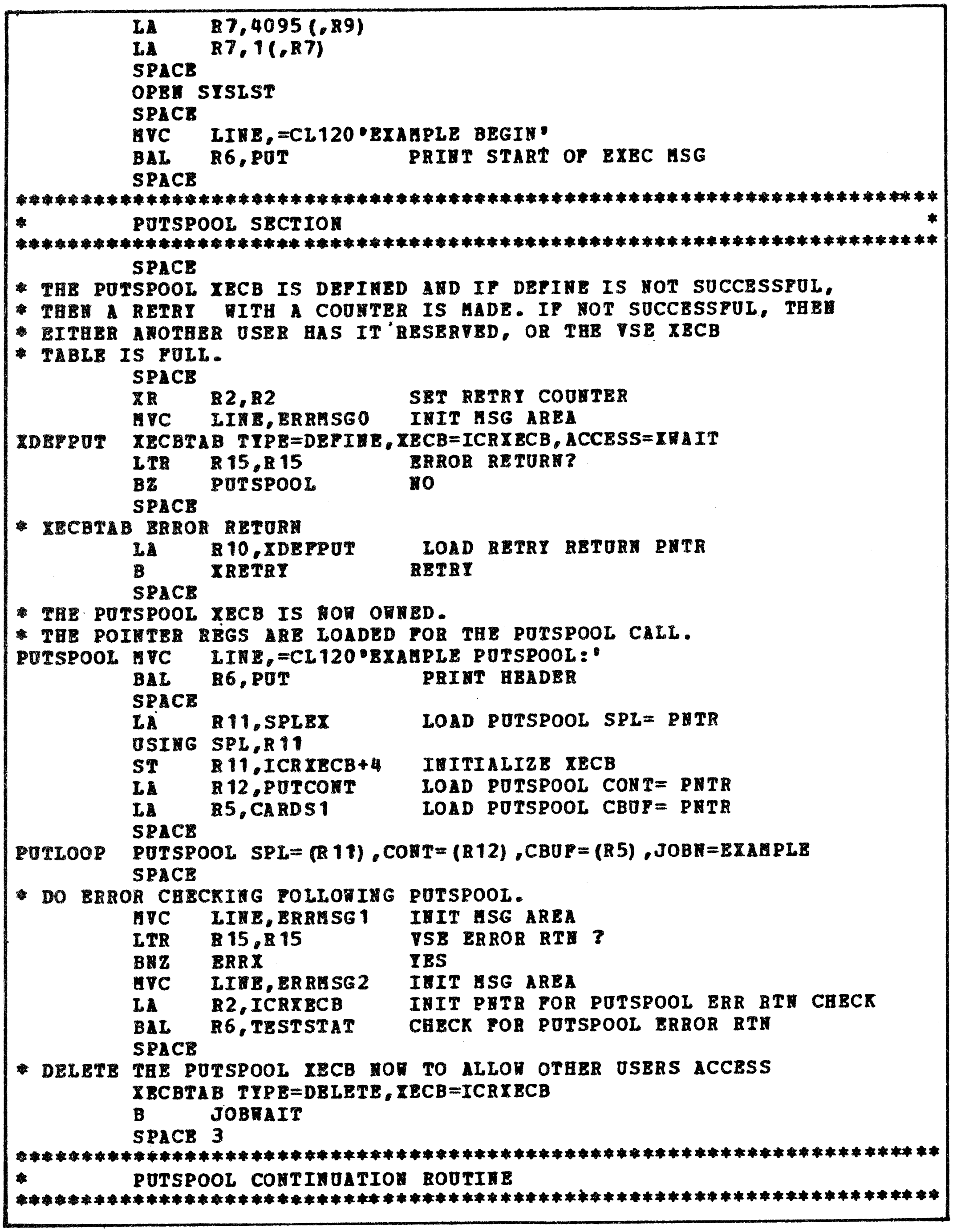
In the example, the F2 partition is used by VSE/POWER, the F3 partition processes the reader CLASS=A input, and the job runs in the BG partition.

Columns 119 and 120 of Figure 65 contain the CCW command code on account of the CC=YES specification in the GETSPOOL macro.



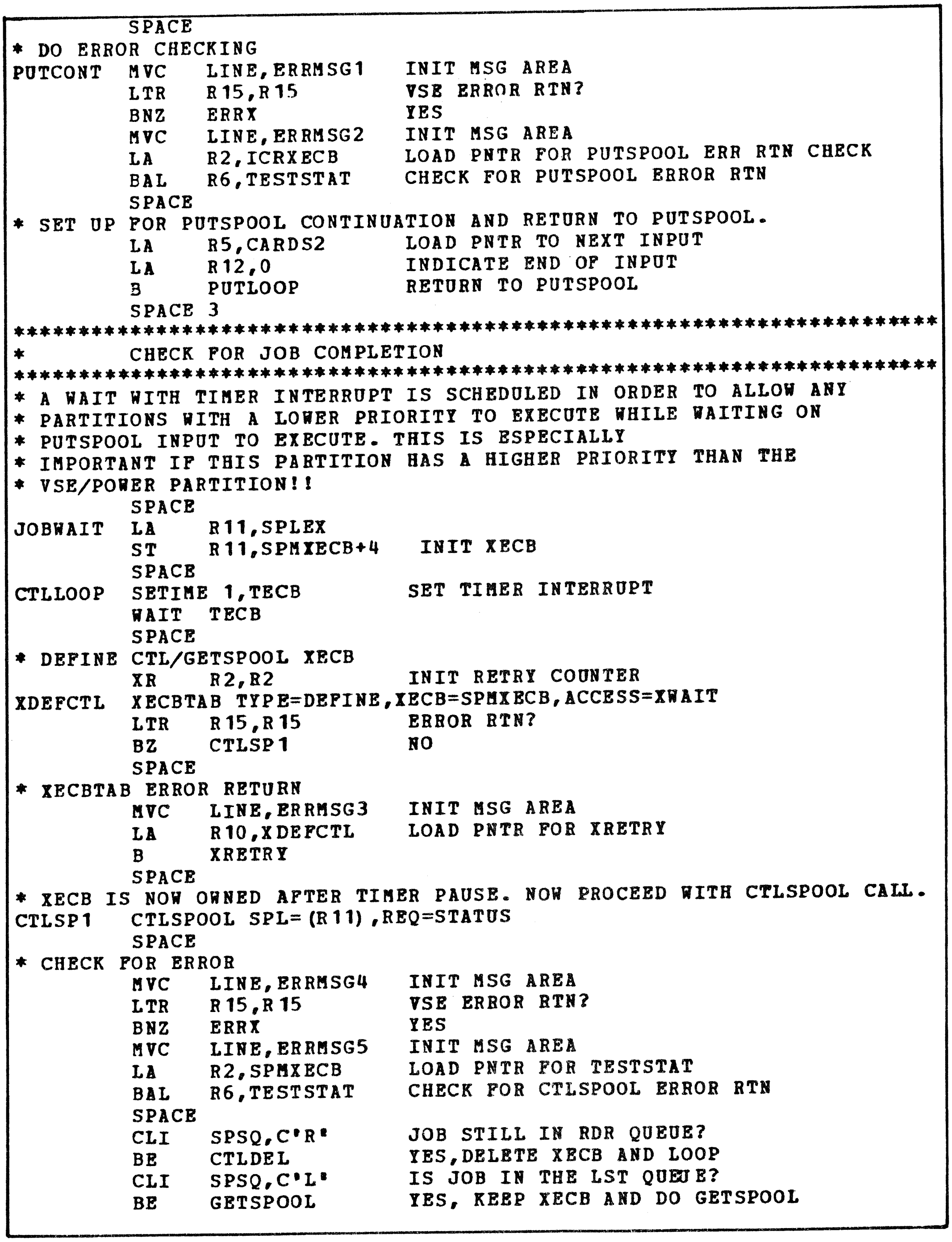
```
         PUNCH   '     PHASE EXAMPLE,S'
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                                     *
*                        V S E / P O W E R                            *
*                                                                     *
*           C R O S S - P A R T I T I O N   E X A M P L E             *
*                                                                     *
*                                                                     *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
         SPACE 3
* THE FOLLOWING IS AN EXAMPLE OF THE USAGE OF VSE/POWER CROSS-PARTITION
* SUPPORT. IT CONFORMS TO THE SUGGESTED PROGRAMMING PRACTICES SO AS
* TO ALLOW A MAXIMUM OF FRICTION-FREE EXISTENCE BETWEEN MULTIPLE USERS.
*
* REGISTER USAGE:
*
*        R0 - VSE  WORK REG (SETIME MACRO)
*        R1 - WORK REGISTER,DOS/VSE  WORK REG (XECBTAB MACRO)
*        R2 - WORK REGISTER
*        R3 - WORK REGISTER
*        R4 - WORK REGISTER
*        R5 - PNTR TO PUTSPOOL CARD INPUT AREA,WORK REGISTER
*        R6 - LINK RETURN REG: PUT,TESTSTAT,HEXCONV ROUTINES
*        R7 - BASE REG 2
*        R8 - GETSPOOL LINENO= PARAMETER VALUE
*        R9 - BASE REG 1
*        R10 - LINK RETURN REG: GETSUB,XRETRY ROUTINES
*        R11 - SPL PNTR
*        R12 - PUTSPOOL CONT= PNTR
*        R13 -
*        R14 - VSE WORK REG (XECBTAB MACRO)
*        R15 - VSE WORK REG
         SPACE 1
R0   EQU    0
R1   EQU    1
R2   EQU    2
R3   EQU    3
R4   EQU    4
R5   EQU    5
R6   EQU    6
R7   EQU    7
R8   EQU    8
R9   EQU    9
R10  EQU    10
R11  EQU    11
R12  EQU    12
R13  EQU    13
R14  EQU    14
R15  EQU    15
         SPACE 3
         EJECT
         PRINT NOGEN
         CSECT
         BALR  R9,0           ESTABLISH ADDRESSABILITY
         USING *,R9,R7
```

Figure 63 (Part 1 of 9). Coding Example for the Use of SPOOL-Macros



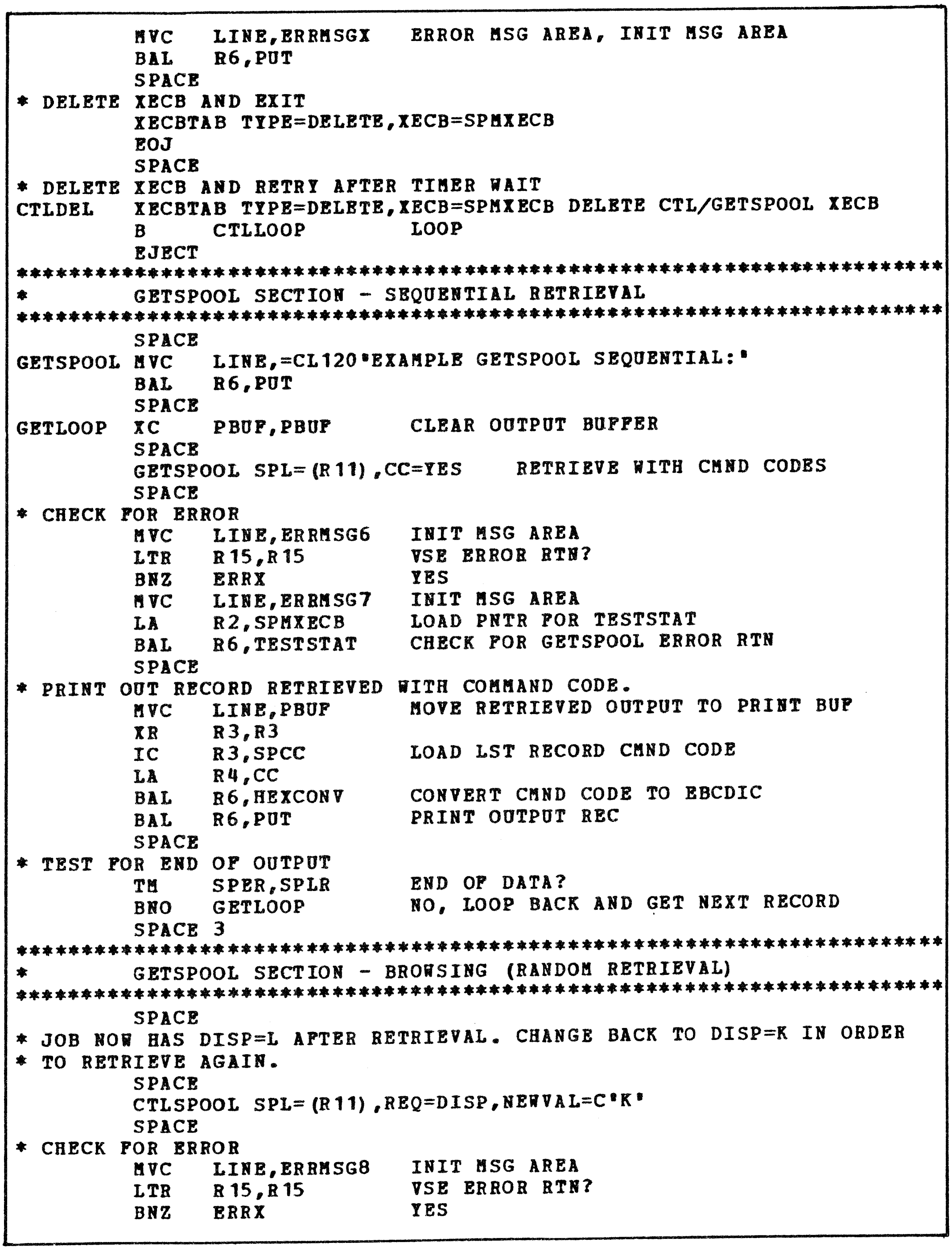
```
         LA    R7,4095(,R9)
         LA    R7,1(,R7)
         SPACE
         OPEN SYSLST
         SPACE
         MVC   LINE,=CL120'EXAMPLE BEGIN'
         BAL   R6,PUT         PRINT START OF EXEC MSG
         SPACE
***********************************************************************
*        PUTSPOOL SECTION                                             *
***********************************************************************
         SPACE
* THE PUTSPOOL XECB IS DEFINED AND IF DEFINE IS NOT SUCCESSFUL,
* THEN A RETRY  WITH A COUNTER IS MADE. IF NOT SUCCESSFUL, THEN
* EITHER ANOTHER USER HAS IT RESERVED, OR THE VSE XECB
* TABLE IS FULL.
         SPACE
         XR    R2,R2          SET RETRY COUNTER
         MVC   LINE,ERRMSG0   INIT MSG AREA
XDEFPUT  XECBTAB TYPE=DEFINE,XECB=ICRXECB,ACCESS=XWAIT
         LTR   R15,R15        ERROR RETURN?
         BZ    PUTSPOOL       NO
         SPACE
* XECBTAB ERROR RETURN
         LA    R10,XDEFPUT     LOAD RETRY RETURN PNTR
         B     XRETRY         RETRY
         SPACE
* THE PUTSPOOL XECB IS NOW OWNED.
* THE POINTER REGS ARE LOADED FOR THE PUTSPOOL CALL.
PUTSPOOL MVC   LINE,=CL120'EXAMPLE PUTSPOOL:'
         BAL   R6,PUT          PRINT HEADER
         SPACE
         LA    R11,SPLEX       LOAD PUTSPOOL SPL= PNTR
         USING SPL,R11
         ST    R11,ICRXECB+4   INITIALIZE XECB
         LA    R12,PUTCONT     LOAD PUTSPOOL CONT= PNTR
         LA    R5,CARDS1       LOAD PUTSPOOL CBUF= PNTR
         SPACE
PUTLOOP  PUTSPOOL SPL=(R11),CONT=(R12),CBUF=(R5),JOBN=EXAMPLE
         SPACE
* DO ERROR CHECKING FOLLOWING PUTSPOOL.
         MVC   LINE,ERRMSG1   INIT MSG AREA
         LTR   R15,R15        VSE ERROR RTN ?
         BNZ   ERRX           YES
         MVC   LINE,ERRMSG2   INIT MSG AREA
         LA    R2,ICRXECB     INIT PNTR FOR PUTSPOOL ERR RTN CHECK
         BAL   R6,TESTSTAT    CHECK FOR PUTSPOOL ERROR RTN
         SPACE
* DELETE THE PUTSPOOL XECB NOW TO ALLOW OTHER USERS ACCESS
         XECBTAB TYPE=DELETE,XECB=ICRXECB
         B     JOBWAIT
         SPACE 3
***********************************************************************
*        PUTSPOOL CONTINUATION ROUTINE
***********************************************************************
```

Figure 63 (Part 2 of 9). Coding Example for the Use of SPOOL-Macros



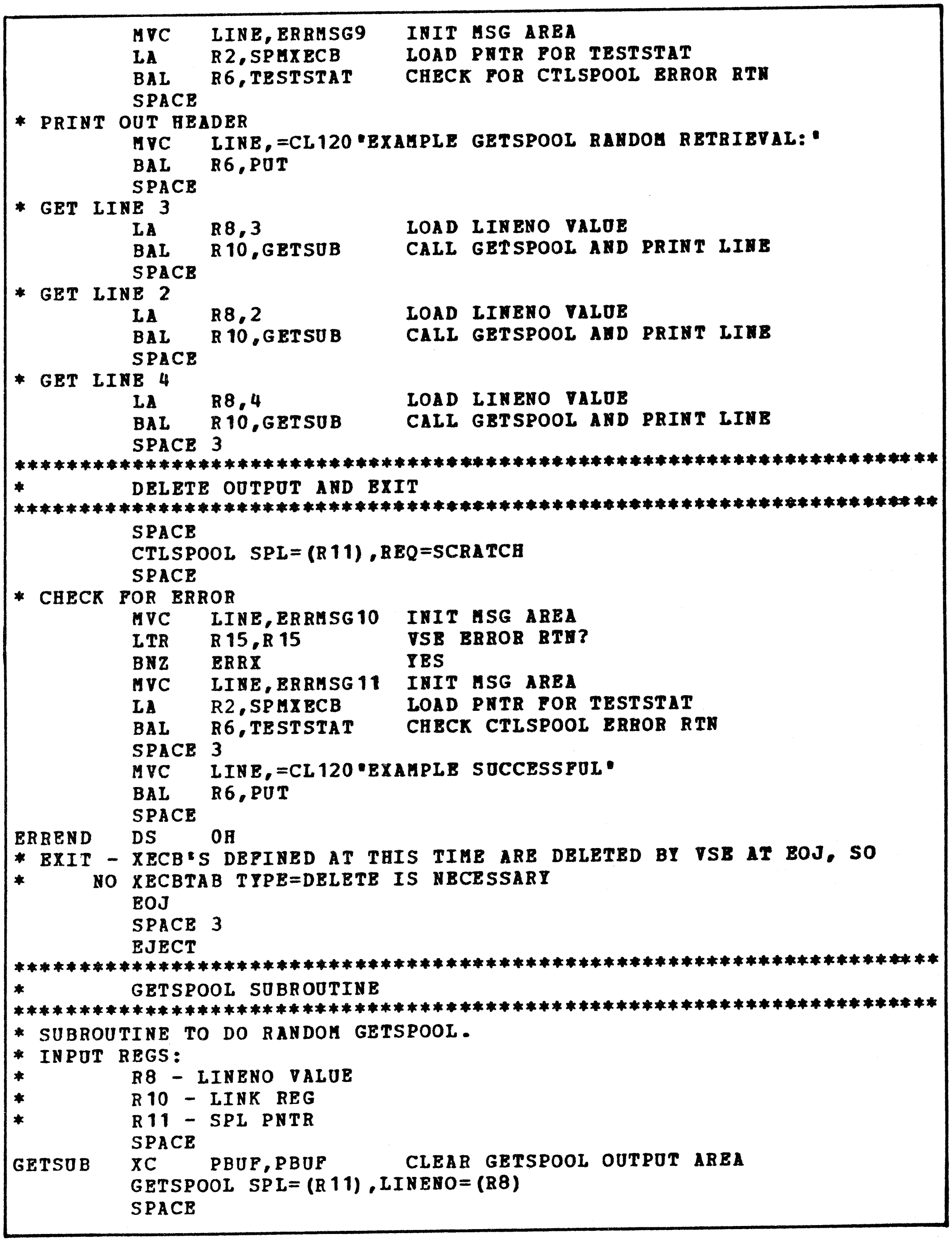
```
         SPACE
* DO ERROR CHECKING
PUTCONT  MVC   LINE,ERRMSG1    INIT MSG AREA
         LTR   R15,R15         VSE ERROR RTN?
         BNZ   ERRX            YES
         MVC   LINE,ERRMSG2    INIT MSG AREA
         LA    R2,ICRXECB      LOAD PNTR FOR PUTSPOOL ERR RTN CHECK
         BAL   R6,TESTSTAT     CHECK FOR PUTSPOOL ERROR RTN
         SPACE
* SET UP FOR PUTSPOOL CONTINUATION AND RETURN TO PUTSPOOL.
         LA    R5,CARDS2       LOAD PNTR TO NEXT INPUT
         LA    R12,0           INDICATE END OF INPUT
         B     PUTLOOP         RETURN TO PUTSPOOL
         SPACE 3
***********************************************************************
*        CHECK FOR JOB COMPLETION
***********************************************************************
* A WAIT WITH TIMER INTERRUPT IS SCHEDULED IN ORDER TO ALLOW ANY
* PARTITIONS WITH A LOWER PRIORITY TO EXECUTE WHILE WAITING ON
* PUTSPOOL INPUT TO EXECUTE. THIS IS ESPECIALLY
* IMPORTANT IF THIS PARTITION HAS A HIGHER PRIORITY THAN THE
* VSE/POWER PARTITION!!
         SPACE
JOBWAIT  LA    R11,SPLEX
         ST    R11,SPMXECB+4    INIT XECB
         SPACE
CTLLOOP  SETIME 1,TECB         SET TIMER INTERRUPT
         WAIT  TECB
         SPACE
* DEFINE CTL/GETSPOOL XECB
         XR    R2,R2           INIT RETRY COUNTER
XDEFCTL  XECBTAB TYPE=DEFINE,XECB=SPMXECB,ACCESS=XWAIT
         LTR   R15,R15         ERROR RTN?
         BZ    CTLSP1          NO
         SPACE
* XECBTAB ERROR RETURN
         MVC   LINE,ERRMSG3    INIT MSG AREA
         LA    R10,XDEFCTL     LOAD PNTR FOR XRETRY
         B     XRETRY
         SPACE
* XECB IS NOW OWNED AFTER TIMER PAUSE. NOW PROCEED WITH CTLSPOOL CALL.
CTLSP1   CTLSPOOL SPL=(R11),REQ=STATUS
         SPACE
* CHECK FOR ERROR
         MVC   LINE,ERRMSG4    INIT MSG AREA
         LTR   R15,R15         VSE ERROR RTN?
         BNZ   ERRX            YES
         MVC   LINE,ERRMSG5    INIT MSG AREA
         LA    R2,SPMXECB      LOAD PNTR FOR TESTSTAT
         BAL   R6,TESTSTAT     CHECK FOR CTLSPOOL ERROR RTN
         SPACE
         CLI   SPSQ,C'R'       JOB STILL IN RDR QUEUE?
         BE    CTLDEL          YES,DELETE XECB AND LOOP
         CLI   SPSQ,C'L'       IS JOB IN THE LST QUEUE?
         BE    GETSPOOL        YES, KEEP XECB AND DO GETSPOOL
```

Figure 63 (Part 3 of 9). Coding Example for the Use of SPOOL-Macros



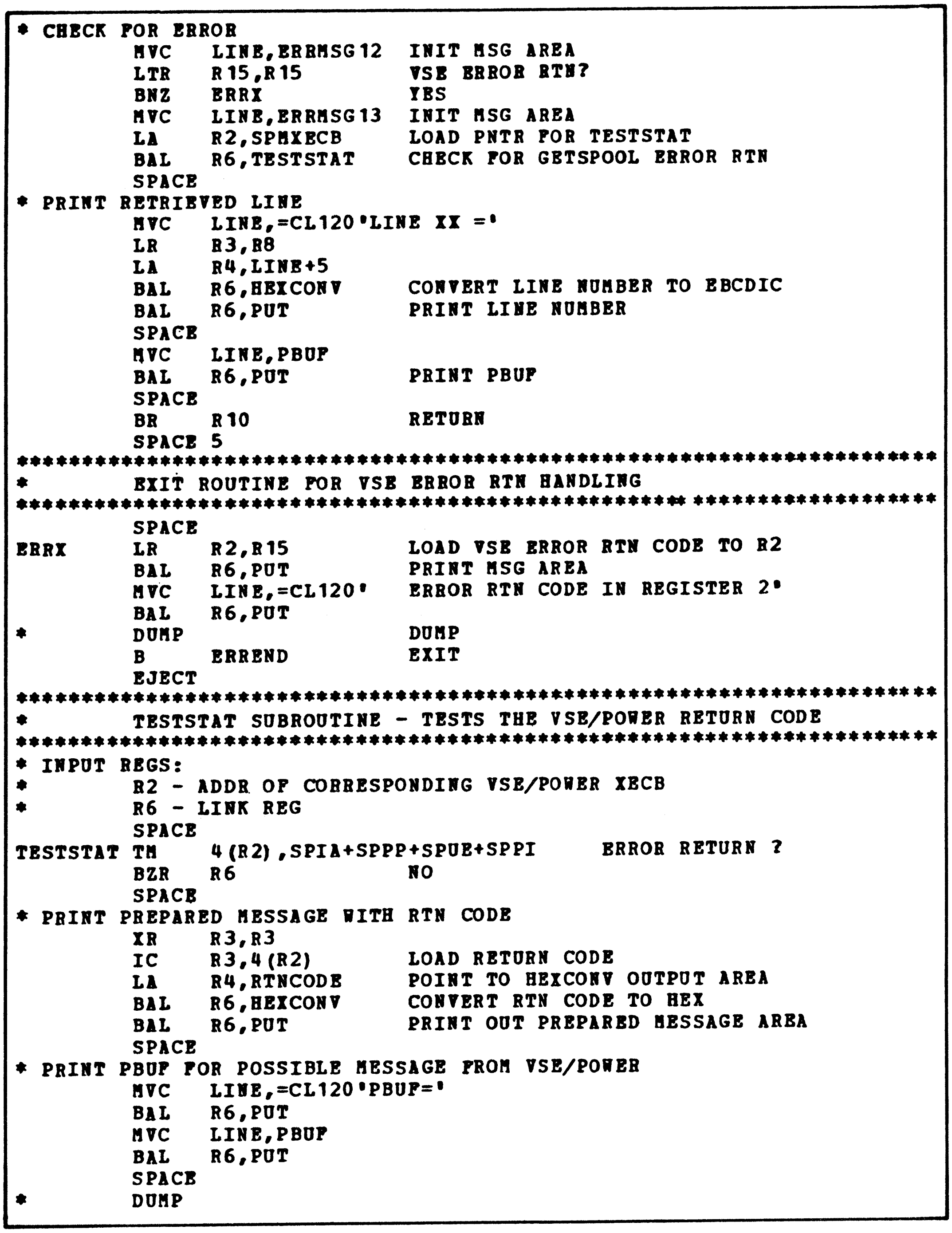
```
         MVC   LINE,ERRMSGX   ERROR MSG AREA, INIT MSG AREA
         BAL   R6,PUT
         SPACE
* DELETE XECB AND EXIT
         XECBTAB TYPE=DELETE,XECB=SPMXECB
         EOJ
         SPACE
* DELETE XECB AND RETRY AFTER TIMER WAIT
CTLDEL   XECBTAB TYPE=DELETE,XECB=SPMXECB DELETE CTL/GETSPOOL XECB
         B     CTLLOOP        LOOP
         EJECT
***********************************************************************
*        GETSPOOL SECTION - SEQUENTIAL RETRIEVAL
***********************************************************************
         SPACE
GETSPOOL MVC   LINE,=CL120'EXAMPLE GETSPOOL SEQUENTIAL:'
         BAL   R6,PUT
         SPACE
GETLOOP  XC    PBUF,PBUF      CLEAR OUTPUT BUFFER
         SPACE
         GETSPOOL SPL=(R11),CC=YES    RETRIEVE WITH CMND CODES
         SPACE
* CHECK FOR ERROR
         MVC   LINE,ERRMSG6   INIT MSG AREA
         LTR   R15,R15        VSE ERROR RTN?
         BNZ   ERRX           YES
         MVC   LINE,ERRMSG7   INIT MSG AREA
         LA    R2,SPMXECB     LOAD PNTR FOR TESTSTAT
         BAL   R6,TESTSTAT    CHECK FOR GETSPOOL ERROR RTN
         SPACE
* PRINT OUT RECORD RETRIEVED WITH COMMAND CODE.
         MVC   LINE,PBUF      MOVE RETRIEVED OUTPUT TO PRINT BUF
         XR    R3,R3
         IC    R3,SPCC        LOAD LST RECORD CMND CODE
         LA    R4,CC
         BAL   R6,HEXCONV     CONVERT CMND CODE TO EBCDIC
         BAL   R6,PUT         PRINT OUTPUT REC
         SPACE
* TEST FOR END OF OUTPUT
         TM    SPER,SPLR      END OF DATA?
         BNO   GETLOOP        NO, LOOP BACK AND GET NEXT RECORD
         SPACE 3
***********************************************************************
*        GETSPOOL SECTION - BROWSING (RANDOM RETRIEVAL)
***********************************************************************
         SPACE
* JOB NOW HAS DISP=L AFTER RETRIEVAL. CHANGE BACK TO DISP=K IN ORDER
* TO RETRIEVE AGAIN.
         SPACE
         CTLSPOOL SPL=(R11),REQ=DISP,NEWVAL=C'K'
         SPACE
* CHECK FOR ERROR
         MVC   LINE,ERRMSG8   INIT MSG AREA
         LTR   R15,R15        VSE ERROR RTN?
         BNZ   ERRX           YES
```

Figure 63 (Part 4 of 9). Coding Example for the Use of SPOOL-Macros



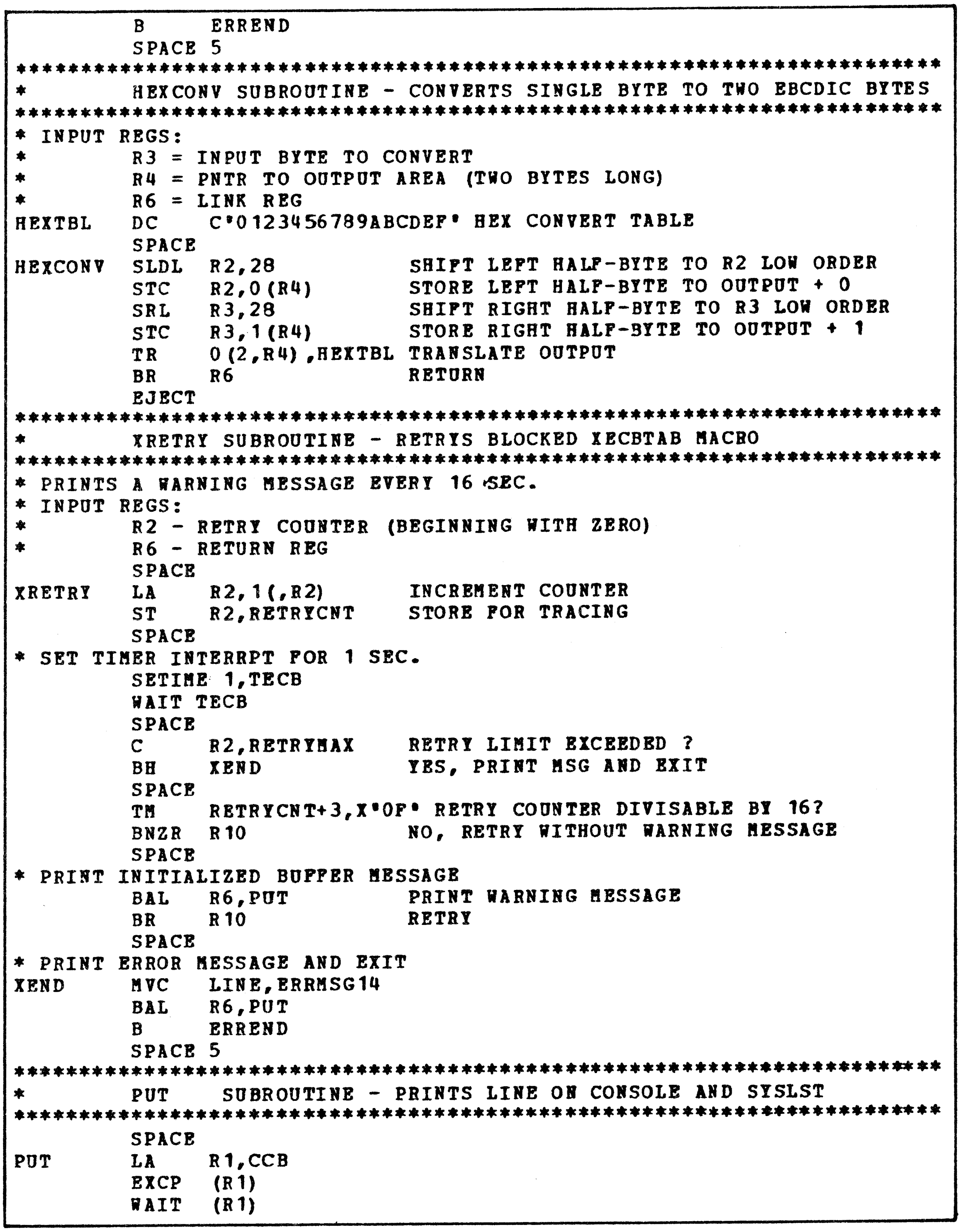
```
         MVC   LINE,ERRMSG9   INIT MSG AREA
         LA    R2,SPMXECB     LOAD PNTR FOR TESTSTAT
         BAL   R6,TESTSTAT    CHECK FOR CTLSPOOL ERROR RTN
         SPACE
* PRINT OUT HEADER
         MVC   LINE,=CL120'EXAMPLE GETSPOOL RANDOM RETRIEVAL:'
         BAL   R6,PUT
         SPACE
* GET LINE 3
         LA    R8,3           LOAD LINENO VALUE
         BAL   R10,GETSUB     CALL GETSPOOL AND PRINT LINE
         SPACE
* GET LINE 2
         LA    R8,2           LOAD LINENO VALUE
         BAL   R10,GETSUB     CALL GETSPOOL AND PRINT LINE
         SPACE
* GET LINE 4
         LA    R8,4           LOAD LINENO VALUE
         BAL   R10,GETSUB     CALL GETSPOOL AND PRINT LINE
         SPACE 3
***********************************************************************
*        DELETE OUTPUT AND EXIT
***********************************************************************
         SPACE
         CTLSPOOL SPL=(R11),REQ=SCRATCH
         SPACE
* CHECK FOR ERROR
         MVC   LINE,ERRMSG10  INIT MSG AREA
         LTR   R15,R15        VSE ERROR RTN?
         BNZ   ERRX           YES
         MVC   LINE,ERRMSG11  INIT MSG AREA
         LA    R2,SPMXECB     LOAD PNTR FOR TESTSTAT
         BAL   R6,TESTSTAT    CHECK CTLSPOOL ERROR RTN
         SPACE 3
         MVC   LINE,=CL120'EXAMPLE SUCCESSFUL'
         BAL   R6,PUT
         SPACE
ERREND   DS    0H
* EXIT - XECB'S DEFINED AT THIS TIME ARE DELETED BY VSE AT EOJ, SO
*     NO XECBTAB TYPE=DELETE IS NECESSARY
         EOJ
         SPACE 3
         EJECT
***********************************************************************
*        GETSPOOL SUBROUTINE
***********************************************************************
* SUBROUTINE TO DO RANDOM GETSPOOL.
* INPUT REGS:
*        R8 - LINENO VALUE
*        R10 - LINK REG
*        R11 - SPL PNTR
         SPACE
GETSUB   XC    PBUF,PBUF      CLEAR GETSPOOL OUTPUT AREA
         GETSPOOL SPL=(R11),LINENO=(R8)
         SPACE
```

Figure 63 (Part 5 of 9). Coding Example for the Use of SPOOL-Macros



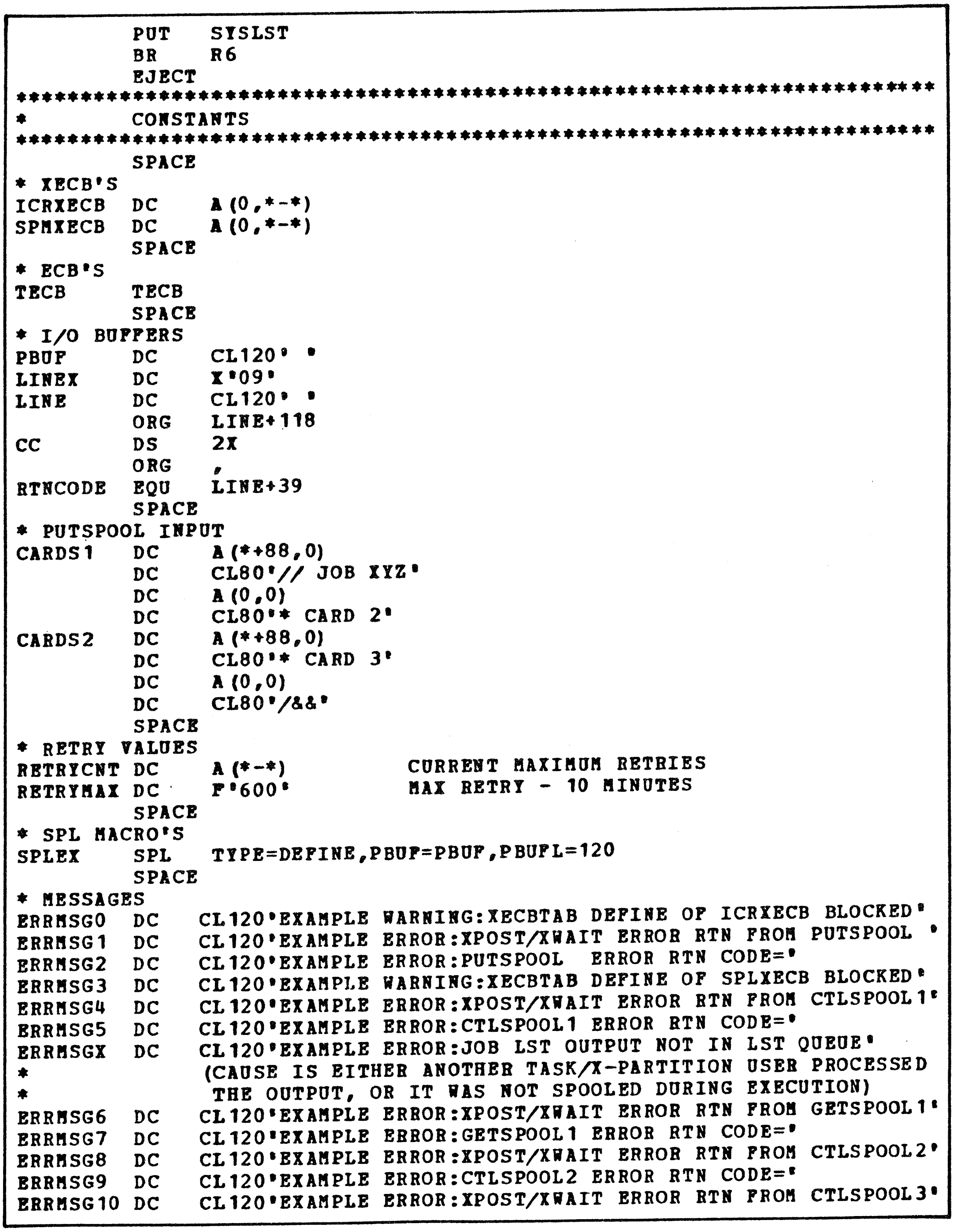
```
* CHECK FOR ERROR
         MVC   LINE,ERRMSG12    INIT MSG AREA
         LTR   R15,R15          VSE ERROR RTN?
         BNZ   ERRX             YES
         MVC   LINE,ERRMSG13    INIT MSG AREA
         LA    R2,SPMXECB       LOAD PNTR FOR TESTSTAT
         BAL   R6,TESTSTAT      CHECK FOR GETSPOOL ERROR RTN
         SPACE
* PRINT RETRIEVED LINE
         MVC   LINE,=CL120'LINE XX ='
         LR    R3,R8
         LA    R4,LINE+5
         BAL   R6,HEXCONV       CONVERT LINE NUMBER TO EBCDIC
         BAL   R6,PUT           PRINT LINE NUMBER
         SPACE
         MVC   LINE,PBUF
         BAL   R6,PUT           PRINT PBUF
         SPACE
         BR    R10              RETURN
         SPACE 5
************************************************************************
*        EXIT ROUTINE FOR VSE ERROR RTN HANDLING
************************************************************************
         SPACE
ERRX     LR    R2,R15           LOAD VSE ERROR RTN CODE TO R2
         BAL   R6,PUT           PRINT MSG AREA
         MVC   LINE,=CL120'     ERROR RTN CODE IN REGISTER 2'
         BAL   R6,PUT
*        DUMP                   DUMP
         B     ERREND           EXIT
         EJECT
************************************************************************
*        TESTSTAT SUBROUTINE - TESTS THE VSE/POWER RETURN CODE
************************************************************************
* INPUT REGS:
*        R2 - ADDR OF CORRESPONDING VSE/POWER XECB
*        R6 - LINK REG
         SPACE
TESTSTAT TM    4(R2),SPIA+SPPP+SPUE+SPPI     ERROR RETURN ?
         BZR   R6               NO
         SPACE
* PRINT PREPARED MESSAGE WITH RTN CODE
         XR    R3,R3
         IC    R3,4(R2)         LOAD RETURN CODE
         LA    R4,RTNCODE       POINT TO HEXCONV OUTPUT AREA
         BAL   R6,HEXCONV       CONVERT RTN CODE TO HEX
         BAL   R6,PUT           PRINT OUT PREPARED MESSAGE AREA
         SPACE
* PRINT PBUF FOR POSSIBLE MESSAGE FROM VSE/POWER
         MVC   LINE,=CL120'PBUF='
         BAL   R6,PUT
         MVC   LINE,PBUF
         BAL   R6,PUT
         SPACE
*        DUMP
```

Figure 63 (Part 6 of 9). Coding Example for the Use of SPOOL-Macros



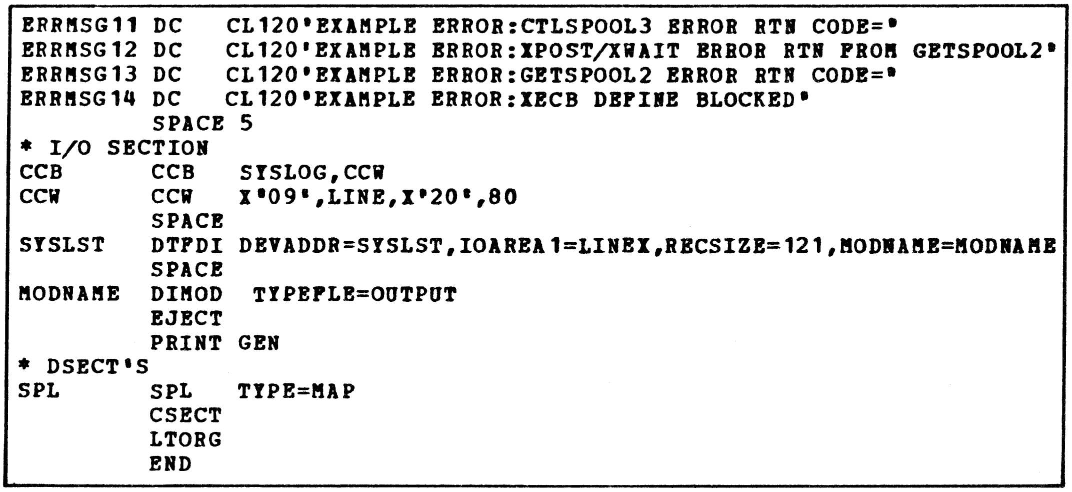
```
         B     ERREND
         SPACE 5
***********************************************************************
*        HEXCONV SUBROUTINE - CONVERTS SINGLE BYTE TO TWO EBCDIC BYTES
***********************************************************************
* INPUT REGS:
*        R3 = INPUT BYTE TO CONVERT
*        R4 = PNTR TO OUTPUT AREA (TWO BYTES LONG)
*        R6 = LINK REG
HEXTBL   DC    C'0123456789ABCDEF' HEX CONVERT TABLE
         SPACE
HEXCONV  SLDL  R2,28           SHIFT LEFT HALF-BYTE TO R2 LOW ORDER
         STC   R2,0(R4)        STORE LEFT HALF-BYTE TO OUTPUT + 0
         SRL   R3,28           SHIFT RIGHT HALF-BYTE TO R3 LOW ORDER
         STC   R3,1(R4)        STORE RIGHT HALF-BYTE TO OUTPUT + 1
         TR    0(2,R4),HEXTBL  TRANSLATE OUTPUT
         BR    R6              RETURN
         EJECT
***********************************************************************
*        XRETRY SUBROUTINE - RETRYS BLOCKED XECBTAB MACRO
***********************************************************************
* PRINTS A WARNING MESSAGE EVERY 16 SEC.
* INPUT REGS:
*        R2 - RETRY COUNTER (BEGINNING WITH ZERO)
*        R6 - RETURN REG
         SPACE
XRETRY   LA    R2,1(,R2)       INCREMENT COUNTER
         ST    R2,RETRYCNT     STORE FOR TRACING
         SPACE
* SET TIMER INTERRPT FOR 1 SEC.
         SETIME 1,TECB
         WAIT TECB
         SPACE
         C     R2,RETRYMAX     RETRY LIMIT EXCEEDED ?
         BH    XEND            YES, PRINT MSG AND EXIT
         SPACE
         TM    RETRYCNT+3,X'OF' RETRY COUNTER DIVISABLE BY 16?
         BNZR  R10             NO, RETRY WITHOUT WARNING MESSAGE
         SPACE
* PRINT INITIALIZED BUFFER MESSAGE
         BAL   R6,PUT          PRINT WARNING MESSAGE
         BR    R10             RETRY
         SPACE
* PRINT ERROR MESSAGE AND EXIT
XEND     MVC   LINE,ERRMSG14
         BAL   R6,PUT
         B     ERREND
         SPACE 5
***********************************************************************
*        PUT    SUBROUTINE - PRINTS LINE ON CONSOLE AND SYSLST
***********************************************************************
         SPACE
PUT      LA    R1,CCB
         EXCP  (R1)
         WAIT  (R1)
```

Figure 63 (Part 7 of 9). Coding Example for the Use of SPOOL-Macros



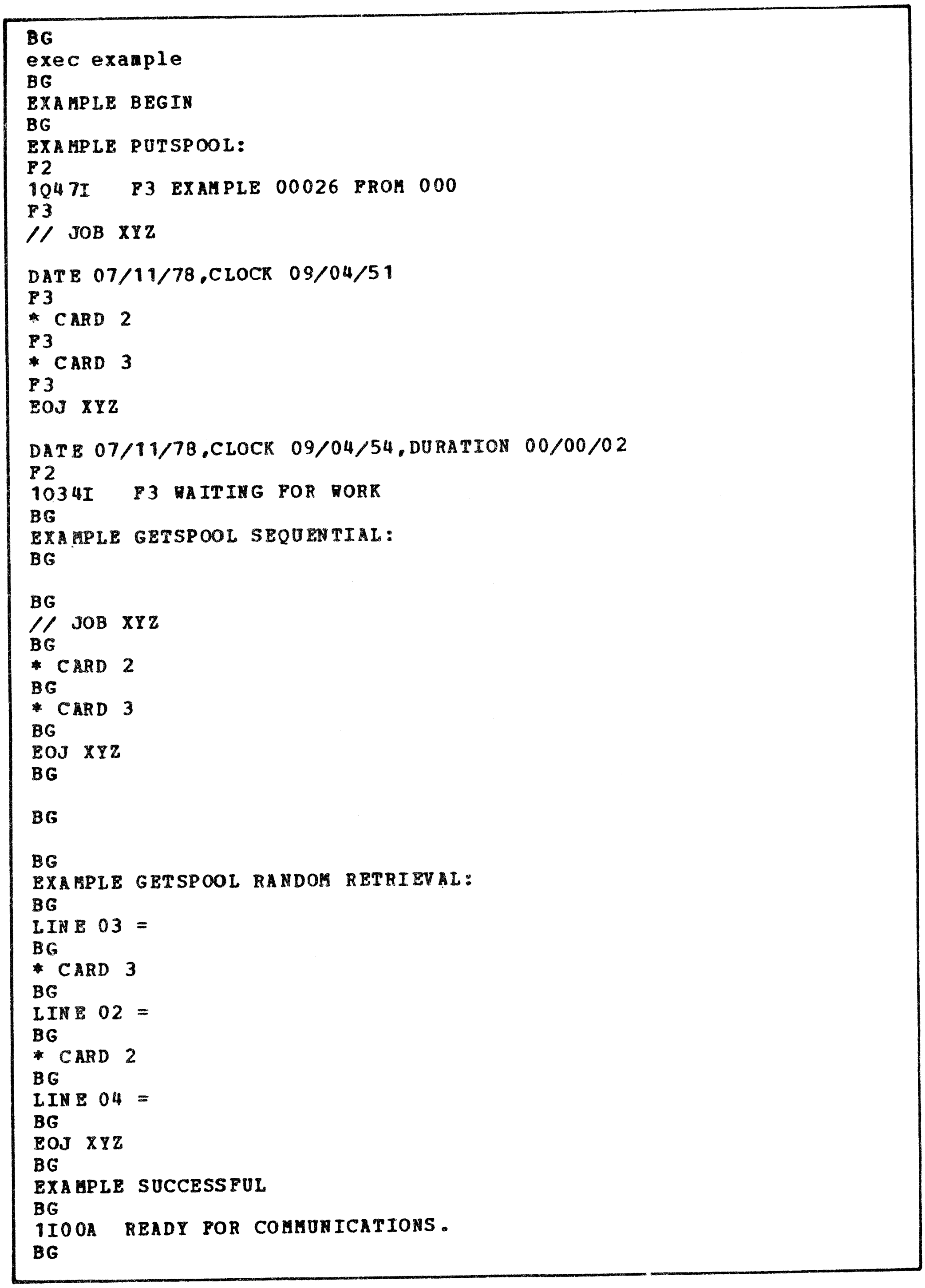
```
         PUT   SYSLST
         BR    R6
         EJECT
***********************************************************************
*        CONSTANTS
***********************************************************************
         SPACE
* XECB'S
ICRXECB  DC    A(0,*-*)
SPMXECB  DC    A(0,*-*)
         SPACE
* ECB'S
TECB     TECB
         SPACE
* I/O BUFFERS
PBUF     DC    CL120' '
LINEX    DC    X'09'
LINE     DC    CL120' '
         ORG   LINE+118
CC       DS    2X
         ORG   ,
RTNCODE  EQU   LINE+39
         SPACE
* PUTSPOOL INPUT
CARDS1   DC    A(*+88,0)
         DC    CL80'// JOB XYZ'
         DC    A(0,0)
         DC    CL80'* CARD 2'
CARDS2   DC    A(*+88,0)
         DC    CL80'* CARD 3'
         DC    A(0,0)
         DC    CL80'/&&'
         SPACE
* RETRY VALUES
RETRYCNT DC    A(*-*)         CURRENT MAXIMUM RETRIES
RETRYMAX DC    F'600'         MAX RETRY - 10 MINUTES
         SPACE
* SPL MACRO'S
SPLEX    SPL   TYPE=DEFINE,PBUF=PBUF,PBUFL=120
         SPACE
* MESSAGES
ERRMSG0  DC   CL120'EXAMPLE WARNING:XECBTAB DEFINE OF ICRXECB BLOCKED'
ERRMSG1  DC   CL120'EXAMPLE ERROR:XPOST/XWAIT ERROR RTN FROM PUTSPOOL '
ERRMSG2  DC   CL120'EXAMPLE ERROR:PUTSPOOL  ERROR RTN CODE='
ERRMSG3  DC   CL120'EXAMPLE WARNING:XECBTAB DEFINE OF SPLXECB BLOCKED'
ERRMSG4  DC   CL120'EXAMPLE ERROR:XPOST/XWAIT ERROR RTN FROM CTLSPOOL1'
ERRMSG5  DC   CL120'EXAMPLE ERROR:CTLSPOOL1 ERROR RTN CODE='
ERRMSGX  DC   CL120'EXAMPLE ERROR:JOB LST OUTPUT NOT IN LST QUEUE'
*             (CAUSE IS EITHER ANOTHER TASK/X-PARTITION USER PROCESSED
*              THE OUTPUT, OR IT WAS NOT SPOOLED DURING EXECUTION)
ERRMSG6  DC   CL120'EXAMPLE ERROR:XPOST/XWAIT ERROR RTN FROM GETSPOOL1'
ERRMSG7  DC   CL120'EXAMPLE ERROR:GETSPOOL1 ERROR RTN CODE='
ERRMSG8  DC   CL120'EXAMPLE ERROR:XPOST/XWAIT ERROR RTN FROM CTLSPOOL2'
ERRMSG9  DC   CL120'EXAMPLE ERROR:CTLSPOOL2 ERROR RTN CODE='
ERRMSG10 DC   CL120'EXAMPLE ERROR:XPOST/XWAIT ERROR RTN FROM CTLSPOOL3'
```

Figure 63 (Part 8 of 9). Coding Example for the Use of SPOOL-Macros



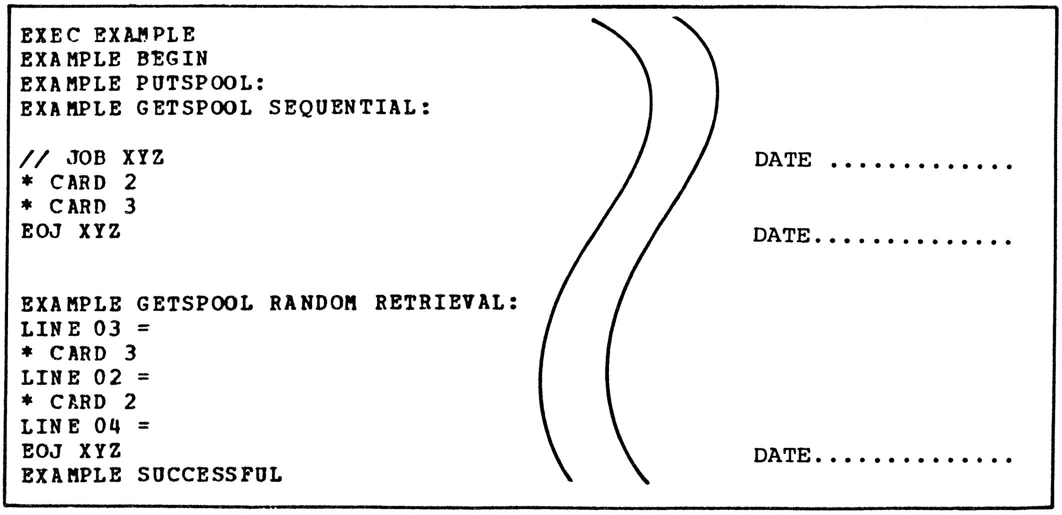
```
ERRMSG11 DC    CL120'EXAMPLE ERROR:CTLSPOOL3 ERROR RTN CODE='
ERRMSG12 DC    CL120'EXAMPLE ERROR:XPOST/XWAIT ERROR RTN FROM GETSPOOL2'
ERRMSG13 DC    CL120'EXAMPLE ERROR:GETSPOOL2 ERROR RTN CODE='
ERRMSG14 DC    CL120'EXAMPLE ERROR:XECB DEFINE BLOCKED'
         SPACE 5
* I/O SECTION
CCB      CCB   SYSLOG,CCW
CCW      CCW   X'09',LINE,X'20',80
         SPACE
SYSLST   DTFDI DEVADDR=SYSLST,IOAREA1=LINEX,RECSIZE=121,MODNAME=MODNAME
         SPACE
MODNAME  DIMOD  TYPEFLE=OUTPUT
         EJECT
         PRINT GEN
* DSECT'S
SPL      SPL   TYPE=MAP
         CSECT
         LTORG
         END
```

Figure 63 (Part 9 of 9). Coding Example for the Use of SPOOL-Macros



```
BG
exec example
BG
EXAMPLE BEGIN
BG
EXAMPLE PUTSPOOL:
F2
1Q47I   F3 EXAMPLE 00026 FROM 000
F3
// JOB XYZ

DATE 07/11/78,CLOCK 09/04/51
F3
* CARD 2
F3
* CARD 3
F3
EOJ XYZ

DATE 07/11/78,CLOCK 09/04/54,DURATION 00/00/02
F2
1034I   F3 WAITING FOR WORK
BG
EXAMPLE GETSPOOL SEQUENTIAL:
BG

BG
// JOB XYZ
BG
* CARD 2
BG
* CARD 3
BG
EOJ XYZ
BG

BG

BG
EXAMPLE GETSPOOL RANDOM RETRIEVAL:
BG
LINE 03 =
BG
* CARD 3
BG
LINE 02 =
BG
* CARD 2
BG
LINE 04 =
BG
EOJ XYZ
BG
EXAMPLE SUCCESSFUL
BG
1I00A  READY FOR COMMUNICATIONS.
BG
```

Figure 64. Console Listing of the SPOOL-Macro Example



```
EXEC EXAMPLE
EXAMPLE BEGIN
EXAMPLE PUTSPOOL:
EXAMPLE GETSPOOL SEQUENTIAL:

// JOB XYZ                                         DATE .............
* CARD 2
* CARD 3
EOJ XYZ                                            DATE..............


EXAMPLE GETSPOOL RANDOM RETRIEVAL:
LINE 03 =
* CARD 3
LINE 02 =
* CARD 2
LINE 04 =
EOJ XYZ                                            DATE..............
EXAMPLE SUCCESSFUL
```

Figure 65. List Output of SPOOL-Macro Example

GLOSSARY

The following terms are defined as they are used in this book. If you do not find the term you are looking for, refer to the index or to the IBM Vocabulary for Data Processing, Telecommunications, and Office Systems, GC20-1699.

This glossary includes definitions developed by the American National Standards Institute (ANSI) and the International Organization for Standardization (ISO). This material is reproduced from the American National Dictionary for Information Processing, (c) 1977 by the Computer and Business Equipment Manufacturers Association, copies of which may be purchased from the American National Standards Institute, 1430 Broadway, New York, New York 10018. Definitions of ANSI are marked with an asterisk (*); definitions of ISO include (ISO) before the definition.

ACB. Access method control block.

account file. A file on disk maintained by VSE/POWER to hold the accounting information collected by VSE/POWER for the programs which it controls.

ACF/VTAM. Advanced Communication Function for the Virtual Telecommunication Access Method. An IBM program product that provides single-domain network capability, and optionally, multiple-domain capability.

adjacent nodes. Two nodes that are connected by a data link with no intervening nodes.

alternate route. The routing path that is used for the transmission of jobs or output from the local node to the destination node if the primary route is not active and signed-on.

*** ASCII.** American National Standard Code for Information Interchange. The standard code, using a coded character set consisting of 7-bit coded characters (8 bits including parity check), used for information interchange among data processing systems, data communication systems, and associated equipment. The ASCII set consists of control characters and graphic characters.

API. Application program identifier.

ASI. (automated system initialization) A method of starting a VSE system with input from cataloged procedures.

autostart. A VSE/POWER function that starts VSE/POWER automatically based on input from SYSIPT or an ASI JCL procedure.

Binary synchronous communication (BSC). Communication using binary synchronous transmission.

BSC. See binary synchronous communication

BSC line. A data transmission line that allows binary synchronous communication (BSC).

channel-to-channel attachment (CTCA). An extension of VSE/POWER's networking function that allows data to be exchanged between two virtual machines under control of VM/System Product.

class. A means of grouping output or jobs that require the same set of resources for their execution.

CMS. Conversational monitor system, a component of VM/SP

cold start. The initialization of input and output work queues. All information present in the queues before the cold start is lost.

CTCA. See channel-to-channel attachment.

data block. The unit of transfer (buffer) that VSE/POWER uses for both disk and tape.

data file. A file on disk, maintained by VSE/POWER to hold the input and output program data records required and generated by VSE programs under control of VSE/POWER.

DBLK (data block) group. The smallest unit of space that can be allocated to a VSE/POWER job on the data file. This allocation is independent of any device characteristics.

device service task. A task of VSE/POWER. The task controls the transfer of output for being printed or punched out by a device owned by a subsystem or program in another partition.

direct link. A direct link is a connection between two adjacent nodes in a network.

disposition. A means of indicating to VSE/POWER how job input and output is to be handled. A job may, for example, be deleted or kept after processing.

drain. To end the transmission of data across a network such that the transmitter (receiver) task being used is no longer available for other work.

Compare also with "flushing".

end node. A node designated as the final destination for a job or output. An end node can be the local or any other node within a network.

execution node. A node at which a job is executed; it is the end node for jobs. An execution node may be another VSE/POWER node or any other node that supports the networking protocol used by VSE/POWER.

flushing. The discontinuation of a job that is being processed by a VSE/POWER task.

intermediate storage. A storage device used by VSE/POWER for spooling job input before execution and job output during execution. In case of spooled output, VSE/POWER uses the device as input the VSE/POWER list and punch tasks. or both.

job entry control language (JECL). A control language that allows the programmer to specify how VSE/POWER is to handle a certain job.

list task. A VSE/POWER task. It controls the writing of spooled job output to a printer. See also "task."

local device. A channel-attached input/output device.

local node. The node at which a user is located and operating. Normally the own VSE system.

multitasking. Concurrent execution of one or more sub-tasks attached to a main task within one partition.

MVS. Multiple virtual storage. A short name for the OS/MVS, an operating system.

name of node. See nodeid.

name of user. See user identifier.

*** network.** An interconnected group of nodes.

networking function. A VSE/POWER function that allows communication between nodes within a network.

Transmission can be via BSC or SDLC lines or CTCA.

networking protocol. A set of records which must be transmitted in a set sequence in order to:

- start a network session
- control the flow of data across through the network
- terminate a network session

The protocol is an agreed standards to which all member nodes in the network must adhere.

node. In a network, an end point of a link or a junction common to two or more links in a network. Nodes can vary in routing and other functional capabilities.

nodeid. Node identifier; the name by which a node is known within a network.

OPTB. See Output Parameter Text Block.

Output Parameter Text Block (OPTB). In VSE/POWER's spool-access support, information that is contained in an output queue record if a * $$ LST or * $$ PUN statement includes any user-defined keywords that have been defined for autostart.

page fault. A program interruption that occurs when a page marked "not in real storage" is referred to by an active page. Synonymous with page translation exception.

PNET. See networking function.

primary route. The preferred routing path out of two such paths that are provided by a direct link for the transmission jobs or job output from the local node to a node with which no direct link exists.

priority. A rank assigned to each job within its class that determines its precedence in receiving system resources.

punch task. A VSE/POWER task. It controls the writing of spooled job output to a punch device. See also "task."

purge. To purge a job or output from the queues means to delete all references to this job or output from VSE/POWER spool files.

queue. A waiting line or list formed by items in a system waiting for service.

queue entry. The queue record of a certain job or output and the related data records, both stored on disk.

queue file. A file on disk, maintained by VSE/POWER to control the processing of user jobs in the system.

queue record. A record in the queue file containing descriptive information about a job or job output.

receiver task. Receives jobs, list and punch output, messages, and commands; writes the received jobs or output to the spool files; queues commands and messages for further transmission or for processing and display, respectively.

remote job entry (RJE) function. Allows jobs to be submitted at RJE work stations for execution by the system and for output to be obtained at a work station.

remote work station. An input/output control unit and one or more input/output devices attached to a system through a transmission control unit.

resource. Any code, control block, table, record, or file used by a task. More properly called "serial resource".

resource management. The mechanisms that protect serial resources from concurrent access by competing tasks.

routing. The process of transmitting jobs and output from one node to another. Routing paths and destinations can be specified in the PNODE generation macro and in JECL statements. See also "alternate route" and "primary route".

RSCS. Remote spooling communications subsystem, a component of VM/SP.

SDLC. See synchronous data link control.

segmentation. A VSE/POWER facility to break bulky list or punch output into segments so that printing or punching can be started before execution of the generating user program has been completed.

session. The period of time during which a user of a remote user can communicate with an interactive system. A session can exist between any two nodes in a network and is established by the use of ACF/VTAM or ACF/VTAME.

shared spooling. A VSE/POWER function that allows two or more VSE systems running under VSE/POWER to share a single set of VSE/POWER spool files and one account file.

spool-access support task. A VSE/POWER task that controls the access of VSE/POWER services for spooling job or output data and for queue manipulation.

spool files. The queue file and data file on disk, used by VSE/POWER for spooling job input and job output.

spooling. The storing of job input and of job output on an auxiliary storage device, concurrently with job execution and in a format convenient for later processing and actual output, again concurrently with job execution.

storage page. A unit of program address space of 2K or 4K bytes and aligned on a 2K or 4K boundary.

synchronous data link control (SDLC). A discipline for managing synchronous, transparent, serial-by-bit information transfer over a communication channel.

task. The basic unit of synchronous program execution. It consists of instructions operating upon program data. Although a task is executed synchronously with respect to its own instructions, these are executed asynchronously with respect to all other tasks existing in the system.

*** transmission.** The sending of data from one place for reception elsewhere.

transmitter task. Controls the transmission, from the local node, of jobs, list and punch output, messages, and commands.

TSO. Time sharing option, a component of MVS

unit of transfer. The amount of virtual storage needed to hold a block of data that is transferred to or from an I/O device by one I/O request.

user identifier. A character string that identifies the user who submitted or is to receive a job or output.

USS. Unformatted system services.

VIO area. See virtual I/O area.

virtual I/O (VIO) area. An area on the page-data-set volume used by system programs.

VM/SP. Virtual Machine/System Product, an IBM program product.

VSE/ICCF. Interactive Computing and Control Facility, an IBM program product.

warm start. A restart that allows reuse of previously initialized input and output work queues. Contrast with cold start.

BIBLIOGRAPHY

Following is a list of IBM publications which you may need to consult. For additional related publications, refer to the latest IBM System/370, 30XX and 4300 Processors Bibliography, GC20-0001.

VSE/POWER:

Installation and Operations Guide, SH12-5329

Remote Job Entry User's Guide, SH12-5328

Networking User's Guide, SC33-6140.

Reference Summary, SH12-5435.

VSE/System Package, Messages and Codes,[1]

SC33-6181 plus SC33-6184 if your VSE/POWER runs on Version 2 of VSE/SP

SC33-6310 if your VSE/POWER runs on Version 3 of VSE/SP

VSE/Advanced Functions:

System Management Guide, SC33-6191.

Planning and Installation, SC33-6193.

System Control Statements, SC33-6198

Operation, SC33-6194.

Application Programming: Macro Users Guide, SC33-6196.

Application Programming: Macro Reference, SC33-6196.

Network Program Products, Planning, SC27-0658

Network Job Entry: Formats and Protocols for System/370 Program Products, GG22-9373.

OS/VS-DOS/VSE-VM/370, Assembler Language, SC33-4010.

DOS/VS IBM 3800 Printing Subsystem Programmer's Guide, GC26-3900.

IBM 3200 Printing Subsystem Support Program, Programmer's Guide, SC18-0100.

[1] All VSE/POWER messages are now contained in Volume 1 of this VSE/SP manaul; the former VSE/POWER Messages manual, SH12-5520 has been discontinued.

INDEX

IBM Virtual Storage Extended/
POWER
Application Programming
Order No. SC33-6276-00

READER'S COMMENT FORM

This form may be used to communicate your comments about this publication, its organization, or subject matter, with the understanding that IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

Your comments will be sent to the author's department for whatever review and action, if any, are deemed appropriate. Comments may be written in your own language; English is not required.

Note: Copies of IBM publications are not stocked at the location to which this form is addressed. Please direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.

Your comments:

Note: **Staples can cause problems with automated mail sorting equipment. Please use pressure sensitive or other gummed tape to seal this form.**

If you wish a reply, give your name and address on the reverse side of this form.

Thank you for your cooperation. No postage stamp is necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments or you may mail directly to the address in the Edition Notice on the back of the title page).

SC33-6276-00

CUT
OR
FOLD
ALONG
LINE

Reader's Comment Form

Fold And Tape **Please Do Not Staple** Fold And Tape

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 40 ARMONK, N.Y.

POSTAGE WILL BE PAID BY ADDRESSEE:

International Business Machines Corporation
Department 6R1BP
180 Kost Road
Mechanicsburg, PA 17055

Fold And Tape **Please Do Not Staple** Fold And Tape

If you would like a reply, please print:

Your Name ____________________

Company Name/Department ____________________

Street Address ____________________

City ____________________

State/Zip Code ____________________

IBM Branch Office serving you ____________________

IBM®

IBM Virtual Storage Extended/
POWER
Application Programming
Order No. SC33-6276-00

READER'S COMMENT FORM

This form may be used to communicate your comments about this publication, its organization, or subject matter, with the understanding that IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

Your comments will be sent to the author's department for whatever review and action, if any, are deemed appropriate. Comments may be written in your own language; English is not required.

Note: Copies of IBM publications are not stocked at the location to which this form is addressed. Please direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.

Your comments:

Note: **Staples can cause problems with automated mail sorting equipment. Please use pressure sensitive or other gummed tape to seal this form.**

If you wish a reply, give your name and address on the reverse side of this form.

Thank you for your cooperation. No postage stamp is necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments or you may mail directly to the address in the Edition Notice on the back of the title page).

SC33-6276-00

CUT
OR
FOLD
ALONG
LINE

Reader's Comment Form

Fold And Tape **Please Do Not Staple** Fold And Tape

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 40 ARMONK, N.Y.

POSTAGE WILL BE PAID BY ADDRESSEE:

International Business Machines Corporation
Department 6R1BP
180 Kost Road
Mechanicsburg, PA 17055

Fold And Tape **Please Do Not Staple** Fold And Tape

If you would like a reply, please print:

Your Name ____________________

Company Name/Department ____________________

Street Address ____________________

City ____________________

State/Zip Code ____________________

IBM Branch Office serving you ____________________

IBM ®

IBM Virtual Storage Extended/
POWER
Application Programming
Order No. SC33-6276-00

READER'S COMMENT FORM

This form may be used to communicate your comments about this publication, its organization, or subject matter, with the understanding that IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

Your comments will be sent to the author's department for whatever review and action, if any, are deemed appropriate. Comments may be written in your own language; English is not required.

Note: Copies of IBM publications are not stocked at the location to which this form is addressed. Please direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.

Your comments:

Note: **Staples can cause problems with automated mail sorting equipment. Please use pressure sensitive or other gummed tape to seal this form.**

If you wish a reply, give your name and address on the reverse side of this form.

Thank you for your cooperation. No postage stamp is necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments or you may mail directly to the address in the Edition Notice on the back of the title page).

SC33-6276-00

CUT
OR
FOLD
ALONG
LINE

Reader's Comment Form

Fold And Tape **Please Do Not Staple** Fold And Tape

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 40 ARMONK, N.Y.

POSTAGE WILL BE PAID BY ADDRESSEE:

International Business Machines Corporation
Department 6R1BP
180 Kost Road
Mechanicsburg, PA 17055

Fold And Tape **Please Do Not Staple** Fold And Tape

If you would like a reply, please print:

Your Name ____________________

Company Name/Department ____________________

Street Address ____________________

City ____________________

State/Zip Code ____________________

IBM Branch Office serving you ____________________

IBM®

SC33-6276-00

IBM Virtual Storage Extended/
POWER
(File No. S370/4300-40)
Printed in U.S.A.