27 VSE

September 1997

update

In this issue

VSE Update

Published by
Xephon
27-35 London Road
Newbury
Berkshire RG14 1JL
England
Telephone: 01635 38342
From USA: 01144 1635 38342
E-mail: xephon@compuserve.com

North American office
Xephon/QNA
1301 West Highway 407, Suite 201-405
Lewisville, TX 75067
USA
Telephone: 940 455 7050

Australian office
Xephon/RSM
GPO Box 6258
Halifax Street
Adelaide, SA 5000
Australia
Telephone: 88 223 1391

Contributions
Articles published in *VSE Update* are paid for at the rate of £170 ($250) per 1000 words for original material. To find out more about contributing an article, without any obligation, please contact us at any of the addresses above and we will send you a copy of our *Notes for Contributors*.

Editor
Fiona Hewitt

Disclaimer
Readers are cautioned that, although the information in this journal is presented in good faith, neither Xephon nor the organizations or individuals that supplied information in this journal give any warranty or make any representations as to the accuracy of the material it contains. Neither Xephon nor the contributing organizations or individuals accept any liability of any kind howsoever arising out of the use of such material. Readers should satisfy themselves as to the correctness and relevance to their circumstances of all advice, information, code, JCL, EXECs, and other contents of this journal before making any use of it.

Subscriptions and back-issues
A year's subscription to *VSE Update*, comprising four quarterly issues, costs £100.00 in the UK, $150.00 in the USA and Canada, £106.00 in Europe, £112.00 in Australasia and Japan, and £110.50 elsewhere. In all cases the price includes postage. Individual issues starting with the March 1991 issue, are available separately to subscribers for £25.00 ($37.50) each including postage.

VSE Update on-line
Code from *VSE Update* can be downloaded from our Web site at http://www.xephon.com; you will need the user-id shown on your address label.

Printed in England.

Spool RDR/LST/PUN queue subroutine

The subroutine presented in this article spools records to the VSE/POWER RDR, LST, or PUN queues passed to it by the calling program in a VSE/SP or VSE/ESA environment.

Two parameters must be passed. These are discussed in turn below.

FIRST PARAMETER

The first parameter must contain two fields.

First field

The first field, the record code, is a one-byte field. It indicates the queue into which the record is to be placed, its length (if spooling to the reader queue), and whether this is the last record being spooled. It is in this field that you can override VSE/POWER's defaults. Additionally, an 'option' record allows the automatic insertion of LDEST=* and/or NTFY=YES operands in the VSE/POWER job statement.

'1' Indicates that the record being passed is an option record with a length of 10 bytes. You may wish to use this specification if your VSE/POWER job statements don't contain LDEST=* and/or NTFY=YES operands, and you wish to insert these operands when spooling to the reader queue. Failure to specify at least an LDEST=* operand will cause VSE/POWER to route job output to the originator (ie DPPOWP); this means that the central site operator must use PALTER to alter the jobs before printing them at the central site. These options remain in effect for the duration of spooling, or until reset by passing another option record. (See the notes below for more information.)

'2' Indicates that a 150-byte RDR set-up record is to be used to override the normal defaults used by VSE/POWER. If this statement is not passed, VSE/POWER uses the defaults established at generation time for the attributes of the queue entry, except for

those speficied on the VSE/POWER JECL job statement. (See the notes below for the layout of this record.)

‘3’ Indicates that the record being passed is to be spooled to the reader queue and that its length is to be 80 bytes.

‘4’ Indicates that the record being passed is to be spooled to the reader queue and that its length is to be 128 bytes.

‘5’ Indicates that a 150-byte PUN set-up record is to be used to override the normal defaults used by VSE/POWER. If this statement is not passed, VSE/POWER uses the defaults established at generation time for the attributes of the queue entry, except for the job name, which is taken from the system communications region of the calling program. (See the notes below for the layout of this record.)

‘6’ Indicates that the record being passed is to be spooled to the PUN queue and that its length is to be 80 bytes. Records spooled to the PUN queue will have a control character of X'01'.

‘7’ Indicates that a 150-byte LST set-up record is to be used to override the normal defaults used by VSE/POWER. If this statement is not passed, VSE/POWER uses the defaults established at generation time for the attributes of the queue entry, except for the job name, which is taken from the system communications region of the calling program. (See the notes below for the layout of this record.)

‘8’ Indicates that the record being passed is to be spooled to the LST queue and that its length is to be 133 bytes. The first byte must contain the control character to be used by VSE/POWER when actual printing is to be performed. Note that VSE/POWER doesn’t check for the validity of this control character – it’s your responsibility to ensure that it is correct. The next 132 bytes contain the actual record to be spooled.

‘9’ Indicates that this is the last record to be spooled. If this code is not passed with the last record, no records will be spooled. If this code is passed before codes 3, 4, 6, or 8, an error will occur. (See the notes below for more information.)

Second field

The second field, the record, must contain the data to be spooled. This record can contain any VSE/POWER JECL or VSE JCL statement or data record. Or, if the record code is a ‘1’, it can contain the values shown in Figure 1.

Position	Value	Meaning
01-01	'Y'	INSERT LDEST=* OPERAND. A value other than 'Y' indicates no insertion.
02-02	'Y'	INSERT NTFY=YES OPERAND. A value other than 'Y' indicates no insertion.
03-10		Filler. Reserved for future use. This field’s length depends on the record code specified in the first field.

Figure 1: Values for second field

SECOND PARAMETER

The second parameter must also contain two fields.

First field

The first field, the return code, is a 5-byte field, and will contain one of the following values on return to the calling program:

- 00000 – The call was successful and there were no errors or messages.
- 11000 – The value of the first field of the first parameter was not 1 through 9.
- 12000 – The value of the first field of the first parameter was ‘9’, but there was no previous request to start spooling (ie you can’t end something you haven’t yet started).
- 13000 – The value of the first and second fields of the first parameter were ‘9* $$ TRM ’, but there was no previous request to start spooling (ie you can’t terminate something you haven’t yet started).

- 15000 – The value of the first field of the first parameter was not consistent with the record code of the first non set-up record passed (ie mixed records have been passed).
- 17000 – The value of the first field of the first parameter contained a 9, but codes 3, 4, 6, or 8 were not passed first.
- 2XXXX – A VSE XPCC 'IDENTIFICATION' error occurred for the reason indicated by the two-byte return code and the two-byte reason code.
- 3XXXX – A VSE XPCC 'CONNECTION' error occurred for the reason indicated by the two-byte return code and the two-byte reason code.
- 4XXXX – Either a VSE XPCC 'CONNECTION ECB' could not be posted for the reason indicated by the two-byte return code and the two-byte reason code, or two minutes have elapsed and it would be futile to wait any longer.
- 5XXXX – A VSE XPCC 'SENDR' error occurred for the reason indicated by the two-byte return code and the two-byte reason code.
- 6XXXX – Either a VSE XPCC 'SENDR ECB' could not be posted for the reason indicated by the two-byte return code and the two-byte reason code, or two minutes have elapsed and it would be futile to wait any longer.
- 7XXXX – A VSE XPCC 'DISCONNECT' error occurred for the reason indicated by the two-byte return code and the two-byte reason code.
- 8XXXX – A VSE XPCC 'TERMINATE' error occurred for the reason indicated by the two-byte return code and the two-byte reason code.
- 9XXXX – The VSE/POWER reply area contains a warning or error message about the job/output just spooled for the reason indicated by the two-byte return code and the two-byte feedback code. This return code is issued when the first or subsequent

buffer has been sent to VSE/POWER (see the notes below for more details). It usually indicates that one or more of the records spooled has an error. Note that in some cases, the job/output may be in the queue with default attributes given to it by VSE/POWER.

- AXXXX – A VSE/POWER 'PWRSPL' error occurred for the reason indicated by the two-byte return code and the two-byte feedback code.
- BXXXX – A VSE/POWER 'PWRSPL' error occurred for the reason indicated by the two-byte return code and the two-byte feedback code.
- CXXXX – A VSE/POWER 'PWRSPL' error occurred for the reason indicated by the two-byte return code and the two-byte feedback code.

  Return codes beginning with 'A', 'B', and 'C' indicate the same type of error. They are different only for diagnostic purposes, to determine in which routine the error occurred. These return codes indicate that VSE/POWER had something to say about the job/output just spooled. This field needn't be cleared before each call.

Second field

The second field, the reply buffer, is a 500-byte field. It contains any messages from VSE/POWER. If there are no messages and an End Of Data (EOD) record was passed, it contains the contents of the spool parameter list (SPL) of the spooled job/output. This field needn't be cleared before each call.

NOTES

1 This subroutine was written and tested under VSE/SP 3.2.3 running VSE/POWER 2.3, and under VSE/ESA 1.3.3 and 1.3.6 running VSE/POWER 5.2.0.

2 If this subroutine is called or LNKEDTED with 'RMODE ANY',

printing of the input records (see below) must not be requested as the CCB/CCW cannot do this. If this restriction is ignored, abends will occur.

3 Before using this subroutine, you should read Chapter 7 of the *VSE/POWER Installation and Operations Guide*, to become familiar with spooling concepts and the use of the XPCC and PWRSPL macros.

4 The records being spooled may be shorter than the length indicated. If this is the case, simply clear the target field to blanks or spaces, and move the record into the correct-length field.

5 DPPOWP uses an internal buffer rather than sending each record passed to it to VSE/POWER. When the buffer is full, the entire contents are sent. When an EOD record is passed (ie the digit '9' in the first field of the first parameter) or any error occurs, the buffer is cleared and reset to its beginning. To terminate the spool request once it has started, without sending any records to VSE/POWER, set the first two fields of the first parameter to '9* $$ TRM ' (without the quotes). This will clear the buffer, reset it to the beginning, and terminate the spool request.

6 It's your responsibility to check the return code after each and every call, and to take any action required.

- If an error occurs, the return code and reply buffer can be displayed/printed to determine the cause of the error. (See the manuals for details on the layout of the reply buffer.)
- If no errors occur, the reply buffer will contain the entire spool parameter list (SPL). The SPL begins with 'SPLDS' and ends with 'SPLGLEN', and contains such fields as the job number of the spooled job/output, its disposition, and so on. See below for details of how to obtain the layout of the SPL.

7 When the option record is used, by specifying a '1' in the first byte of the first field of the first parameter, or when the RDRGPW or RDRDTNN fields of the RDR set-up record are greater than

spaces, additional statements are inserted when submitting to the RDR queue. For example, if you indicate to insert the LDEST=* parameter, the following occurs:

Before job is submitted:

```
* $$ JOB JNM=...
* $$ LST CLASS=...
// JOB ...
... REMAINDER OF JOB ...
```

After job is submitted:

```
* $$ JOB JNM=...,                                                    X
* $$ LDEST=*
* $$ LST CLASS=...
// JOB ...
... REMAINDER OF JOB ...
```

In the above example, a comma has been inserted after the last operand, a continuation character has been inserted in column 72 of the job statement, and the LDEST=* statement has been inserted.

In the following example, the LDEST=* statement has been inserted and a comma and a continuation character have been placed in the LDEST=* statement as the job statement is continued.

Before job is submitted:

```
* $$ JOB JNM=...,                                                    X
* $$ USER=PAYROLL
* $$ LST CLASS=...
// JOB ...
... REMAINDER OF JOB ...
*
```

After job is submitted:

```
* $$ JOB JNM=...,                                                    X
* $$ LDEST=*,                                                        X
* $$ USER=PAYROLL
* $$ LST CLASS=...
// JOB ...
... REMAINDER OF JOB ...
```

If the RDRGPW field of the RDR set-up record is greater than spaces, the following occurs:

Before job is submitted:

```
* $$ JOB JNM=...
* $$ LST CLASS=...
// JOB ...
... REMAINDER OF JOB ...
```

After job is submitted:

```
* $$ JOB JNM=...,                                                X
* $$ PWD=...
* $$ LST CLASS=...
// JOB ...
... REMAINDER OF JOB ...
```

There is no check to determine whether the LDEST=*, NTFY=YES, PWD=, or XDEST= operands have already been specified. There is commented code that does this, but it checks for these operands only in the first job statement, not in continued job statements. Testing has shown that VSE/POWER takes the last specification when operands are duplicated.

8 Note that when the option record is used and you have indicated to insert the NTFY=YES operand, you must also set the RDRGUS field in the RDR set-up record with a valid ICCF userid so that VSE/POWER can notify the user of job status.

9 When spooling a job, the first record should be a VSE/POWER * $$ JOB JECL statement, as this is where VSE/POWER finds the attributes (class, disposition, etc). If there isn't one, VSE/POWER uses the job name from the VSE // JOB statement if present. Otherwise, 'autoname' and the default attributes are used. Also, the job should end with a VSE/POWER EOJ statement (ie '* $$ EOJ '). If not, one will be inserted to avoid an error condition.

10 When spooling records to any queue, you should end each spool request with an EOD record – VSE/POWER collects all error messages in the reply buffer, and it may therefore be difficult to determine which one caused an error when more than one entry

is spooled. Additionally, VSE/POWER doesn't recognize job boundaries for LST/PUN output. This means that if the records being passed are as follows, only one queue entry is created:

```
6PUNCH RECORD 1  JOB 1
6PUNCH RECORD 2  JOB 1
6PUNCH RECORD 3  JOB 1
6PUNCH RECORD 1  JOB 2
6PUNCH RECORD 2  JOB 2
9PUNCH RECORD 3  JOB 3
```

Finally, you shouldn't intermix RDR/LST/PUN records, as the queue to which the records are to be spooled is determined by the first non set-up record passed (ie codes 3, 4, 6, or 8). For example:

```
3* $$ JOB JNM=....
3* $$ LST CLASS=....
3// JOB ...
3...ADDITIONAL JECL, JCL OR DATA STATEMENTS...
4...ADDITIONAL JECL, JCL OR DATA STATEMENTS...
3...ADDITIONAL JECL, JCL OR DATA STATEMENTS...
3/&
9* $$ EOJ
7LIST SETUP RECORD
8LIST RECORD 1
8...ADDITIONAL LIST RECORDS...
9* $$ EOD
```

Note that:

- The above example illustrates the mixing of 80- and 128-byte records, which is permissible. Both VSE and VSE/POWER ignore positions 81-128 for JCL and JECL statements.
- In the above example, it would be permissible to spool additional records to the RDR queue after spooling to the LST queue.

11 When spooling records to any queue, you may not know when you've read the last record until after it's been passed. For instance, the calling program may be reading records from disk. An End Of File (EOF) won't be recognized until the next READ/GET is issued. If this is the case, set the first two fields of the first parameter to one of the following (without the quotes):

‘9* $$ EOD ’

‘9* $$ CTL ’

This will send the buffer to VSE/POWER and terminate the spool request.

12 As a debugging aid, you may print, to the device assigned to SYSLST, all incoming records passed to this subroutine. You do this by placing an X'FE' in the second byte of the first field of the second parameter (the return code). If you wish to print all records, you must move the X'FE' to the second byte of the return code for each and every call.

13 The VSE XPCC return/reason codes are listed in the manuals. If you can’t find the return/reason code, your manual may not be current, or IBM may have added new codes without updating the manual. You can print the return/reason codes by executing the following JCL (insert your own VSE/POWER JECL and VSE JCL):

```
// EXEC ASSEMBLY,SIZE=256K
      MAPXPCCB
/*
```

The return code is named ‘IJBXRETC’ and the reason code is named ‘IJBXREAS’. The error identification follows each.

14 The VSE/POWER return/feedback codes are listed in the manuals. If you can’t find the return/feedback code, your manual may not be current, or IBM may have added new codes without updating the manual. You can print the return/feedback codes by executing the following JCL (insert your own VSE/POWER JECL and VSE JCL):

```
// EXEC ASSEMBLY,SIZE=256K
      PWRSPL TYPE=MAP
/*
```

The return/feedback codes follow the DSECT ‘PXPUSER’. The return code is named ‘PXPRETCD’ and the feedback code is named ‘PXPFBKCD’. The error identification follows each.

15 The RDR/PUN/LST queue set-up records allow you to override the defaults used by VSE/POWER when creating queue entries. If used, they must precede the queue records being spooled, and are in effect until an EOD record is passed or an error occurs. Subsequent set-up records passed before the EOD record are ignored. If you pass back-to-back set-up records before the first data record, the last one will be used.

Because any field greater than low-values is passed to VSE/POWER, its contents must be valid for VSE/POWER's use – if not, an appropriate return/feedback code is issued. This means that the types of data contained within the fields (alpha, alphanumeric, numeric) and its range values must be correct. All alphanumeric data must be left-justified and padded on the right with blanks. Except for the RDR fields, those indicated by an asterisk (*) allow the use of an ampersand (&) in the first position so that the equivalent SPL field will be set to spaces.

		* VALID FOR *		
POSITION	FIELD NAME	RDR	LST	PUN
001-001	RECORD CODE. (2=RDR,5=PUN,7=LST)	YES	YES	YES
002-009	JOB NAME.	NO	YES	YES
010-010	OUTPUT CLASS. (A-Z)	NO	YES	YES
* 011-018	PASSWORD.	NO	YES	YES
019-019	OUTPUT DISPOSITION. (D,H,K,L,N)	NO	YES	YES
020-020	OUTPUT PRIORITY. (0-9)	NO	YES	YES
021-021	OUTPUT SYSTEM IDENTIFIER. (1-9,N)	NO	YES	YES
* 022-037	USER INFORMATION.	NO	YES	YES
* 038-045	NAME OF DESTINATION NODE.	NO	YES	YES
* 046-053	NAME OF DESTINATION USER.	NO	YES	YES
054-073	PROGRAMMER NAME.	YES	YES	YES
074-081	ROOM NUMBER.	YES	YES	YES
082-089	DEPARTMENT NUMBER.	YES	YES	YES
090-097	BUILDING NUMBER.	YES	YES	YES
098-099	NUMBER OF COPIES. (01-99)	N/A	YES	YES
100-103	FILLER.	N/A	N/A	N/A
* 104-107	FORM NUMBER.	N/A	YES	YES
108-111	FILLER.	N/A	N/A	N/A
112-119	EXTERNAL WRITER.	N/A	YES	N/A
* 120-127	FCB IMAGE PHASE. (IE. FCB NAME)	N/A	YES	N/A
* 128-135	UCB IMAGE PHASE. (IE. UCB NAME)	N/A	YES	N/A
136-136	UCB OPTIONS. (F=FOLD,C=CHECK, B=BOTH)	N/A	YES	N/A
137-138	SEPARATOR PAGES/CARDS. (00-99)	N/A	YES	YES

```
139-140   FILLER.                                N/A  N/A  N/A
* 141-148 ORIGINATING USER ID                    YES  YES  YES
149-149   COMMAND CODE. (A=ASA, M=MACHINE)       N/A  YES  N/A
150-150   CONVERT ASA TO MACHINE. (Y=YES,        N/A  YES  N/A
           N=NO)
```

Note that, except for the last two, the fields pertain to possible entries on VSE/POWER JECL statements. The last two fields are associated only with an LST queue entry, and have the following functions:

– Command code. If an ASA control character is contained in the first byte of the 133-byte LST queue record being spooled, you must inform VSE/POWER by inserting an 'A' into this field. If the value entered is not an 'A', 'M' (machine code) is assumed.

– Convert ASA. If the previous field (command code) indicated an 'A', you can tell VSE/POWER to convert the ASA control character(s) into machine code. This has not been put to any formal test, but VSE/POWER is supposed to be able to handle this.

Fields for the 3800 are not supported.

CALLING SEQUENCES

The calling sequences are given below.

COBOL

```
CALL 'DPPOWP' USING PARAM1, PARAM2.
```

ALC

```
LA    13,SAVEAREA (13 CAN ALSO BE R13 OR RD).
                  CALL  DPPOWP,(PARAM1,PARAM2)
                  .
                  .  (MAINLINE PART OF PROGRAM).
                  .
SAVEAREA DC    18F'0'
```

RPGII

```
     CALL 'DPPOWP'
                  PARM           PARAM1
                  PARM           PARAM2
```

An 18-word save area must be passed through register 13 by the user (STD COBOL LINKAGE).

```
     DFHCSAD
     COPY  DFHCSADS
     DFHTCA CICSYST=YES        COPY DFHTCADS.
```

DPPOWP

```
         TITLE 'DPPOWP - 1.Ø - VSE/POWER SPOOL RDR/LST/PUN QUEUE
               SUBROUTINE'
DPPOWP   CSECT Ø
DPPOWP   AMODE 31
DPPOWP   RMODE 24
         SAVE  (14,12)             SVE REGS.
         BALR  2,Ø                 LOAD BASE REG.
         USING *,2,9               INFORM ASSEMBLER.
         LA    9,4Ø95(2)           LOAD SECOND BASE REG WITH
         LA    9,1(9)                 CONTENTS OF FIRST +4Ø96.
         ST    13,SAVEAREA+4       STORE CALLERS RETURN ADDRESS.
         LA    13,SAVEAREA         LOAD CALLERS REGS.
         B     POWBEG              BRANCH TO POWBEG.
*
         DC    C'DPPOWP STARTS HERE. ' INSERT EYE CATCHER.
*
POWBEG   EQU   *
         MVI   POWPRTR,C' '        CLEAR PRINT RECORD.
         MVC   POWPRTR+1(L'POWPRTR-1),POWPRTR ...
         LM    7,8,Ø(1)            GET ADDRESSES OF PASSED
                                   PARAMETERS.
         MVC   POWCDE,Ø(8)         MVE PASSED RETURN CODE.
         MVC   POWCDS,POWCDE       SVE RETURN CODE.
*        MVI   POWCDS+1,X'FE'      FORCE PRINT INCOMING RECORDS.
         MVC   POWCDE,=C'ØØØØØ'    ASSUME CALL WILL BE SUCCESSFUL.
         CLI   Ø(7),C'1'           IS CODE LOWER THAN '1'.
         BL    POWBEG7             YES-BRANCH TO POWBEG7.
         CLI   Ø(7),C'9'           IS CODE HIGHER THAN '9'.
         BNH   POWBEG9             NO-BRANCH TO POWBEG9.
*
POWBEG7  EQU   *
         MVI   POWCDE,C'1'         INDICATE RECORD CODE ERROR.
         MVI   POWCDE+1,C'1'       ...
```

```
         B     POWEXT              BRANCH TO POWEXT.
*
POWBEG9  EQU   *
         CLI   Ø(7),C'1'           IS THIS A OPTION RECORD.
         BNE   POWSKØ              NO-BRANCH TO POWSKØ.
         MVC   OPTREC,Ø(7)         MVE OPTION RECORD TO SVE AREA.
         MVC   OPTSAV,Ø(7)         MVE OPTION RECORD TO SVE AREA.
         MVC   POWPRTR(11),Ø(7)    MVE OPTION RECORD TO PRINT RECORD.
         BAL   1Ø,POWPRT           PERFORM POWPRT ROUTINE.
         B     POWEXT              BRANCH TO POWEXT.
*
POWSKØ   EQU   *
         CLI   Ø(7),C'2'           IS THIS AN RDR QUEUE SETUP RECORD.
         BNE   POWSK1              NO-BRANCH TO POWSK1.
         MVC   RDRREC,Ø(7)         MVE RDR SETUP RECORD TO SVE AREA.
         MVC   POWPRTR,Ø(7)        MVE RDR SETUP RECORD TO PRINT
                                   RECORD
         BAL   1Ø,POWPRT           PERFORM POWPRT ROUTINE.
         B     POWEXT              BRANCH TO POWEXT.
*
POWSK1   EQU   *
         CLI   Ø(7),C'5'           IS THIS A PUN QUEUE SETUP RECORD.
         BNE   POWSK3              NO-BRANCH TO POWSK3.
         MVC   PUNREC,Ø(7)         MVE PUN SETUP RECORD TO SVE AREA.
         MVC   POWPRTR,Ø(7)        MVE PUN SETUP RECORD TO PRINT RECORD
         BAL   1Ø,POWPRT           PERFORM POWPRT ROUTINE.
         B     POWEXT              BRANCH TO POWEXT.
*
POWSK3   EQU   *
         CLI   Ø(7),C'7'           IS THIS AN LST QUEUE SETUP RECORD.
         BNE   POWSK5              NO-BRANCH TO POWSK5.
         MVC   LSTREC,Ø(7)         MVE LST SETUP RECORD TO SVE AREA.
         MVC   POWPRTR,Ø(7)        MVE LST SETUP RECORD TO PRINT RECORD
         BAL   1Ø,POWPRT           PERFORM POWPRT ROUTINE.
         B     POWEXT              BRANCH TO POWEXT.
*
POWSK5   EQU   *
         LA    1Ø,POWREP           LOAD ADDRESS OF POWREP TO REG 1Ø.
         LA    11,L'POWREP         LOAD LENGTH OF POWREP TO REG 11.
         L     15,=X'4Ø000ØØØ'     SET PAD BYTE AND LENGTH FOR MVCL.
         MVCL  1Ø,14               CLEAR IT.
         MVI   POWREC,C' '         CLEAR POWREC AREA.
         MVC   POWREC+1(149),POWREC ...
         LA    3,XPCCB
         USING IJBXPCCB,3
         LA    4,IJBXSUSR
         USING PXUUSER,4
         LA    5,IJBXRUSR
```

```
         USING PXPUSER,5
         LA    6,OWNSPL
         USING OWNSPLDS,6
         MVC   POWREC,Ø(7)         MVE PASSED RECORD TO SVE AREA (81)
         CLI   POWREC,C'9'         ARE WE ENDING SPOOLING.
         BNE   POWSK6              NO-BRANCH TO POWSK6.
         CLI   POWSSW,C'1'         ARE WE IN PROCESS OF SPOOLING.
         BE    POWSK6              YES-BRANCH TO POWSK6.
         MVI   POWCDE,C'1'         INDICATE END ERROR.
         MVI   POWCDE+1,C'2'       ...
         B     POWEXT              BRANCH TO POWEXT.
*
POWSK6   EQU   *
         CLC   POWREC(L'POWTRMR),POWTRMR IS THIS A TERMINATE RECORD.
         BNE   POWSK7              NO-BRANCH TO POWSK7.
         MVC   POWPRTR,Ø(7)        MVE TERMINATE RECORD TO PRINT RECORD
         BAL   1Ø,POWPRT           PERFORM POWPRT ROUTINE.
         BAL   14,POWCSB           PERFORM POWCSB ROUTINE.
         CLI   POWSSW,C'1'         ARE WE IN PROCESS OF SPOOLING.
         MVC   POWSSW(4),=C'ØØØØØ' CLEAR SWITCHES.
         BE    POWDIS              YES-BRANCH TO POWDIS.
         MVI   POWCDE,C'1'         INDICATE TRM ERROR.
         MVI   POWCDE+1,C'3'       ...
         B     POWEXT              BRANCH TO POWEXT.
*
POWSK7   EQU   *
         CLI   POWREC,C'4'         ARE WE SPOOLING 128 BYTE RECORDS.
         BNE   *+14                NO-SKIP NEXT TWO (2) INST.
         MVI   POWRSW,C'1'         INDICATE SPOOLING TO RDR QUEUE.
         MVC   POWREC(L'POWREC+48),Ø(7) MVE PASSED RECORD TO SVE AREA.
         CLI   POWREC,C'8'         ARE WE SPOOLING 133 BYTE RECORDS.
         BNE   *+14                NO-SKIP NEXT TWO (2) INST.
         MVI   POWLSW,C'1'         INDICATE SPOOLING TO LST QUEUE.
         MVC   POWREC(L'POWREC+53),Ø(7) MVE PASSED RECORD TO SVE AREA.
         CLI   POWREC,C'3'         ARE WE SPOOLING TO RDR QUEUE.
         BNE   *+8                 NO-SKIP NEXT INST.
         MVI   POWRSW,C'1'         INDICATE SPOOLING TO RDR QUEUE.
         CLI   POWREC,C'6'         ARE WE SPOOLING TO PUN QUEUE.
         BNE   *+8                 NO-SKIP NEXT INST.
         MVI   POWPSW,C'1'         INDICATE SPOOLING TO PUN QUEUE.
         CLC   POWRSW(3),=C'1ØØ'   ARE RECORDS MIXED.
         BE    POWSK8              NO-BRANCH TO POWSK8.
         CLC   POWRSW(3),=C'Ø1Ø'   ARE RECORDS MIXED.
         BE    POWSK8              NO-BRANCH TO POWSK8.
         CLC   POWRSW(3),=C'ØØ1'   ARE RECORDS MIXED.
         BE    POWSK8              NO-BRANCH TO POWSK8.
         MVI   POWCDE,C'1'         INDICATE MIX ERROR.
         MVI   POWCDE+1,C'5'       ...
         BAL   14,POWCSB           PERFORM POWCSB ROUTINE.
```

```
         MVC   POWSSW(4),=C'ØØØØØ' CLEAR SWITCHES.
         B     POWDIS              BRANCH TO POWDIS.
*
POWSK8   EQU   *
         CLI   POWSSW,C'1'         ARE WE IN PROCESS OF SPOOLING.
         BE    POWFIL              YES-BRANCH TO POWFIL.
         CLI   POWREC,C'9'         IS THIS END OF DATA RECORD.
         BNE   POWSK8C             NO-BRANCH TO POWSK8C.
         MVI   POWCDE,C'1'         INDICATE EOD ERROR.
         MVI   POWCDE+1,C'7'       ...
         B     POWEXT              BRANCH TO POWEXT.
*
POWSK8C  EQU   *
         EXTRACT ID=CPUID,AREA=POWCPU,LEN=L'POWCPU
         LTR   15,15               WAS EXTRACT SUCCESSFUL.
         BZ    POWSK8F             YES-BRANCH TO POWSK8F.
         MVC   POWPID,=C'FF'
*
POWSK8F  EQU   *
         AMODESW QRY ,             QUERY AMODE.
         STCM  1,B'1ØØØ',MODE      SVE IT.
         MVC   USERID(6),=C'DPPOWP' MVE 'DPPOWP' TO USERID.
         MVC   USERID+6(L'POWPID),POWPID MVE PARTITION ID TO USERID.
         ASYSCOM (1)               GET SYSTEM COMMUNICATIONS ADDRESS.
         USING SYSCOM,1            INFORM ASSEMBLER.
         MVC   SIJBPIK,IJBTIK      SVE TIK. (X'5A').
         DROP  1                   (SYSCOM).
         COMRG                     GET COMMUNICATIONS REGION.
         USING COMREG,1
         MVC   JOBNME,COMNAME      SVE JOB NAME. (X'18').
         MVC   PGMNME,IJBPHNAM     SVE EXECUTING PROGRAM NAME. (X'D8').
         L     1Ø,IJBAFCB          LOAD ADDRESS OF IJBAFCB. (X'B4').
         L     11,8(1Ø)            LOAD POSSIBLE ADDRESS OF CSA (X'Ø8')
         ST    11,ADDRCSA          SVE IT.
         DROP  1                   (COMREG).
         MVI   POWCSW,C'Ø'         INDICATE NOT RUNNING UNDER ICCF/CICS
*        CLC   SIJBPIK,=X'ØØ2F'    ARE WE RUNNING UNDER AN ATTACHED PHA
*        BH    POWIDT              YES-BRANCH TO POWIDT.
         GETFLD FIELD=ICCFPP       GET INTERACTIVE PARTITION FLAG.
         LTR   1,1                 ARE WE RUNNING IN ICCF PSEUDO PARTIT
         BNZ   POWIDT              YES-BRANCH TO POWIDT.
*        CLC   =C'DPATCH',PGMNME   ARE WE RUNNING UNDER CICS.
*        BE    POWSK9              YES-BRANCH TO POWSK9.
         CLC   =C'DTSINIT',PGMNME  ARE WE RUNNING UNDER ICCF/CICS.
         BE    POWSK9              YES-BRANCH TO POWSK9.
         CLC   =C'DFHSIP',PGMNME   ARE WE RUNNING UNDER CICS.
         BNE   POWIDT              NO-BRANCH TO POWIDT.
*
POWSK9   EQU   *
```

```
         STM   Ø,15,SVØØ15
         MVI   POWCSW,C'1'         INDICATE RUNNING UNDER ICCF/CICS.
         USING DFHCSADS,13
         L     13,ADDRCSA          ADDRESS CSA.
         TM    CSASSI2,CSAPLTPI    ARE WE IN CICS STARTUP.
         BNO   POWIDT              YES-BRANCH TO POWIDT.
         MVC   USERID,BLANKS       CLEAR USERID.
         L     12,CSACDTA          ADDRESS CURRENT DISPATCHED USER TCA.
         LR    1Ø,12               LOAD SYSTEM TCA AREA ADDRESS TO REG
         L     14,Ø(1Ø)            LOAD ADDRESS OF SYSTEM TCA TO REG 14
         USING DFHTCADY,14         INFORM ASSEMBLER.
         L     15,TCATCUCN         LOAD ADDRESS OF TCTTE USER CHAIN ADD
         MVC   USERID(4),Ø(15)     MVE TERMINAL ID TO USERID.
         OC    USERID(4),BLANKS
         MVC   TERMID,USERID
         LA    1Ø,USERID
         LA    1,4
*
POWSK9A  EQU   *
         CLI   Ø(1Ø),C'A'
         BL    POWSK9B
         CLI   Ø(1Ø),C'9'
         BH    POWSK9B
*
POWSK9AA EQU   *
         LA    1Ø,1(1Ø)
         BCT   1,POWSK9A
         B     POWSK9D
*
POWSK9B  EQU   *
         MVI   Ø(1Ø),C'@'
         B     POWSK9AA
*
POWSK9D  EQU   *
         LA    1Ø,USERID-1         LOAD ADDRESS OF USERID-1 TO REG 1Ø.
*
POWSK9E  EQU   *
         LA    1Ø,1(1Ø)            INCREMENT TO NEXT POSITION.
         CLI   Ø(1Ø),C' '          IS THIS POSITION BLANK.
         BNE   POWSK9E             NO-BRANCH TO POWSK9E.
         CLI   37(15),C'A'
         BL    POWSK9H
         CLI   37(15),C'9'
         BH    POWSK9H
         CLC   =X'ØØØØØØ',37(15)   IS OPERATOR SIGNED ON. (X'25').
         BE    POWSK9H             NO-BRANCH TO POWSK9G.
         MVC   Ø(3,1Ø),37(15)      MVE OPERATOR ID TO USERID.
*
POWSK9G  EQU   *
```

```
         LA    1Ø,1(1Ø)            INCREMENT TO NEXT POSITION.
*
POWSK9H  EQU   *
         CLI   Ø(1Ø),C' '          IS THIS POSITION BLANK.
         BNE   POWSK9G             NO-BRANCH TO POWSK9G.
*
POWSK9J  EQU   *
         MVC   Ø(L'POWPID-1,1Ø),POWPID+1 MVE LAST BYTE OF PARTITION ID
         LM    Ø,15,SVØØ15
         DROP  13,14
         CLI   MODE,X'8Ø'          ARE WE RUNNING ABOVE THE LINE.
         BNE   POWIDT              NO-BRANCH TO POWIDT.
         MVI   POWCSW,C'Ø'         INDICATE NOT RUNNING UNDER ICCF/CICS
*
POWIDT   EQU   *
         MVC   IJBXAPPL(L'USERID),USERID MVE USERID TO XPCCB.
         XPCC  XPCCB=(3),FUNC=IDENT IDENTIFY THYSELF.
         CLM   15,1,X'Ø8'          WAS RETURN CODE X'Ø8'.
         BNO   POWCON              NO-BRANCH TO POWCON.
         MVI   POWCDE,C'2'         INDICATE XPCC IDENTIFICATION FAILURE
         BAL   1Ø,POWXRC           PERFORM POWXRC ROUTINE.
         B     POWRTN              BRANCH TO POWRTN.
*
POWCON   EQU   *
         XPCC  XPCCB=(3),FUNC=CONNECT CONNECT THYSELF.
         LTR   15,15               WAS CONNECTION ALREADY AVAILABLE.
         BZ    POWCLR              YES-BRANCH TO POWCLR.
         CLM   15,1,X'Ø8'          WAS RETURN CODE X'Ø8'.
         BL    POWWAI              NO-BRANCH TO POWWAI. (MUST BE X'Ø4')
         MVI   POWCDE,C'3'         INDICATE XPCC CONNECTION FAILURE.
         BAL   1Ø,POWXRC           PERFORM POWXRC ROUTINE.
         B     POWRTN              BRANCH TO POWRTN.
*
POWWAI   EQU   *
         CLI   POWCSW,C'1'         ARE WE RUNNING UNDER ICCF/CICS.
         BE    POWWAIC3            YES-BRANCH TO POWWAIC3.
         SETIME 12Ø,INTECB         SET WAIT INTERVAL TO TWO MINUTES.
         LA    1Ø,IJBXCECB         LOAD ADDRESS OF CONNECTION ECB.
         ST    1Ø,LISTCECB         COMPLETE WAITLIST.
         WAITM WAITLIST            WAIT FOR CONNECTION OR 2 MINUTES.
         TM    IJBXCECB+2,X'8Ø'    CONNECTION COMPLETE.
         BO    POWCLR              YES-BRANCH TO POWCLR.
*
POWWAIB3 EQU   *
         CLI   POWCSW,C'Ø'         ARE WE RUNNING UNDER ICCF/CICS.
         BE    POWWAIB5            NO-BRANCH TO POWWAIB5.
         AP    POWCNT,=P'1'        ADD ONE (1) TO WAIT COUNTER.
         CP    POWCNT,=P'15'       HAVE WE TRIED 15 TIMES.
         BL    POWWAIC5            NO-BRANCH TO POWWAIC5.
```

```
*
POWWAIB5 EQU   *
         MVI   POWCDE,C'4'         INDICATE XPCC CONNECTION ERROR.
         BAL   1Ø,POWXRC           PERFORM POWXRC ROUTINE.
         B     POWRTN              BRANCH TO POWRTN.
*
POWWAIC3 EQU   *
         SP    POWCNT,POWCNT       CLEAR COUNTER.
*
POWWAIC5 EQU   *
         STM   Ø,15,SVØØ15
         USING DFHCSADS,13
         L     13,ADDRCSA          ADDRESS CSA.
         L     12,CSACDTA          ADDRESS CURRENT DISPATCHED USER TCA.
         LA    11,IJBXCECB         LOAD ADDRESS OF CONNECTION ECB TO RE
         ST    11,TCATCEA          STORE IT IN TCATCEA.
         DFHKC TYPE=WAIT,DCI=SINGLE
         LM    Ø,15,SVØØ15
         DROP  13
         B     POWWAIB3            BRANCH TO POWWAIB3.
*
POWCLR   EQU   *
         LA    1Ø,REPLBUF          LOAD ADDRESS OF REPLBUF TO REG 1Ø.
         LA    11,L'REPLBUF        LOAD LENGTH OF REPLBUF TO REG 11.
         L     15,=X'4ØØØØØØØ'     SET PAD BYTE AND LENGTH FOR MVCL.
         MVCL  1Ø,14               CLEAR IT.
         CLI   POWREC,C'6'         ARE WE SPOOLING TO PUN QUEUE.
         BE    POWPU1              YES-BRANCH TO POWPU1.
         CLI   POWREC,C'8'         ARE WE SPOOLING TO LST QUEUE.
         BE    POWLS1              YES-BRANCH TO POWLS1.
*        MVC   POWPWD,=8X'ØØ'      CLEAR PASSWORD.
         MVC   POWPWD,=8C' '       CLEAR PASSWORD.
         CLI   POWCSW,C'1'         ARE WE RUNNING UNDER ICCF/CICS.
         BE    POWCLR3             YES-BRANCH TO POWCLR3.
         PWRSPL TYPE=UPD,SPL=OWNSPL,QUEUE=RDR,REQ=PUT,
               JOBN=JOBNME,PWD=POWPWD,USERID=USERID
         B     POWCLR5             BRANCH TO POWCLR5.
*
POWCLR3  EQU   *
         PWRSPL TYPE=UPD,SPL=OWNSPL,QUEUE=RDR,REQ=PUT,OPT=NOWAIT,
               JOBN=JOBNME,PWD=POWPWD,USERID=USERID
*
POWCLR5  EQU   *
         MVC   SPLDLREC,H128       SET MAXIMUM RECORD LENGTH TO 128.
*        CLI   RDRJOB,X'Ø1'        IS JOB NAME LOW-VALUE.
*        BL    *+1Ø                YES-SKIP NEXT INST.
*        MVC   SPLGJB,RDRJOB       REPLACE IT.
*        CLI   RDRGCL,X'Ø1'        IS INPUT CLASS LOW-VALUE.
*        BL    *+1Ø                YES-SKIP NEXT INST.
```

```
*        MVC   SPLGCL,RDRGCL       REPLACE IT.
         CLI   RDRGPW,X'Ø1'        IS PASSWORD LOW-VALUE.
         BL    *+1Ø                YES-SKIP NEXT INST.
         MVC   SPLGPW,RDRGPW       REPLACE IT.
*        CLI   RDRDDP,X'Ø1'        IS INPUT DISPOSITION LOW-VALUE.
*        BL    *+1Ø                YES-SKIP NEXT INST.
*        MVC   SPLDDP,RDRDDP       REPLACE IT.
*        CLI   RDRDPR,X'Ø1'        IS PRIORITY LOW-VALUE.
*        BL    *+1Ø                YES-SKIP NEXT INST.
*        MVC   SPLDPR,RDRDPR       REPLACE IT.
*        CLI   RDRDSID,X'Ø1'       IS INPUT SYSTEM IDENTIFIER LOW VALUE
*        BL    *+1Ø                YES-SKIP NEXT INST.
*        MVC   SPLDSID,RDRDSID     REPLACE IT.
*        CLI   RDRDUI,X'Ø1'        IS USER INFORMATION LOW-VALUE.
*        BL    *+1Ø                YES-SKIP NEXT INST.
*        MVC   SPLDUI,RDRDUI       REPLACE IT.
         MVC   SPLDTNN,BLANKS
*        CLI   RDRDTNN,X'Ø1'       IS NAME OF DESTINATION NODE LOW VALUE
*        BL    *+1Ø                YES-SKIP NEXT INST.
*        MVC   SPLDTNN,RDRDTNN     REPLACE IT.
         MVC   SPLDTUID,BLANKS
*        CLI   RDRDTUID,X'Ø1'      IS NAME OF DESTINATION USER LOW VALUE
*        BL    *+1Ø                YES-SKIP NEXT INST.
*        MVC   SPLDTUID,RDRDTUID   REPLACE IT.
         CLI   RDRDPRGN,X'Ø1'      IS PROGRAMMER NAME LOW-VALUE.
         BL    *+1Ø                YES-SKIP NEXT INST.
         MVC   SPLDPRGN,RDRDPRGN   REPLACE IT.
         CLI   RDRDROOM,X'Ø1'      IS ROOM NUMBER LOW-VALUE.
         BL    *+1Ø                YES-SKIP NEXT INST.
         MVC   SPLDROOM,RDRDROOM   REPLACE IT.
         CLI   RDRDDEPT,X'Ø1'      IS DEPARTMENT NUMBER LOW-VALUE.
         BL    *+1Ø                YES-SKIP NEXT INST.
         MVC   SPLDDEPT,RDRDDEPT   REPLACE IT.
         CLI   RDRDBLDG,X'Ø1'      IS BUILDING NUMBER LOW-VALUE.
         BL    *+1Ø                YES-SKIP NEXT INST.
         MVC   SPLDBLDG,RDRDBLDG   REPLACE IT.
         CLI   RDRGUS,X'41'        IS ORIGINATING USER ID HEXZERO/BLANK
         BL    POWNXT              YES-BRANCH TO POWNXT.
         MVC   SPLGUS,RDRGUS       REPLACE IT.
         B     POWNXT              BRANCH TO POWNXT.
*
POWPU1   EQU   *
*        XC    POWPWD,POWPWD       CLEAR PASSWORD.
         MVC   POWPWD,=8C' '       CLEAR PASSWORD.
         PWRSPL TYPE=UPD,SPL=OWNSPL,QUEUE=PUN,REQ=PUT,
               JOBN=JOBNME,PWD=POWPWD,USERID=USERID
         MVC   SPLDLREC,HØ8Ø       SET MAXIMUM RECORD LENGTH TO 8Ø.
         MVI   SPLORCFM,SPLORMCC   INDICATE RECORD FORMAT IS MACH CODE.
         CLI   PUNJOB,X'Ø1'        IS JOB NAME HEXZERO/BLANK.
```

```
BL    *+1Ø                YES-SKIP NEXT INST.
MVC   SPLGJB,PUNJOB       REPLACE IT.
CLI   PUNGCL,X'Ø1'        IS OUTPUT CLASS LOW-VALUE.
BL    *+1Ø                YES-SKIP NEXT INST.
MVC   SPLGCL,PUNGCL       REPLACE IT.
CLI   PUNGPW,X'Ø1'        IS PASSWORD LOW-VALUE.
BL    *+1Ø                YES-SKIP NEXT INST.
MVC   SPLGPW,PUNGPW       REPLACE IT.
CLI   PUNGPW,C'&&'        IS PASSWORD AN AMPERSAND (&).
BNE   *+1Ø                NO-SKIP NEXT INST.
MVC   SPLGPW,BLANKS       CLEAR IT.
CLI   PUNDDP,X'Ø1'        IS OUTPUT DISPOSITION LOW-VALUE.
BL    *+1Ø                YES-SKIP NEXT INST.
MVC   SPLDDP,PUNDDP       REPLACE IT.
CLI   PUNDPR,X'Ø1'        IS PRIORITY LOW-VALUE.
BL    *+1Ø                YES-SKIP NEXT INST.
MVC   SPLDPR,PUNDPR       REPLACE IT.
CLI   PUNDSID,X'Ø1'       IS OUTPUT SYSTEM IDENTIFIER LOW VALUE
BL    *+1Ø                YES-SKIP NEXT INST.
MVC   SPLDSID,PUNDSID     REPLACE IT.
CLI   PUNDUI,X'Ø1'        IS USER INFORMATION LOW-VALUE.
BL    *+1Ø                YES-SKIP NEXT INST.
MVC   SPLDUI,PUNDUI       REPLACE IT.
CLI   PUNDUI,C'&&'        IS USER INFORMATION AN AMPERSAND (&)
BNE   *+1Ø                NO-SKIP NEXT INST.
MVC   SPLDUI,BLANKS       CLEAR IT.
MVC   SPLDTNN,BLANKS
CLI   PUNDTNN,X'Ø1'       IS NAME OF DESTINATION NODE LOW VALUE
BL    *+1Ø                YES-SKIP NEXT INST.
MVC   SPLDTNN,PUNDTNN     REPLACE IT.
CLI   PUNDTNN,C'&&'       IS NAME OF DESTINATION NODE AN AMPER
BNE   *+1Ø                NO-SKIP NEXT INST.
MVC   SPLDTNN,BLANKS      CLEAR IT.
MVC   SPLDTUID,BLANKS
CLI   PUNDTUID,X'Ø1'      IS NAME OF DESTINATION USER LOW VALU
BL    *+1Ø                YES-SKIP NEXT INST.
MVC   SPLDTUID,PUNDTUID   REPLACE IT.
CLI   PUNDTUID,C'&&'      IS NAME OF DESTINATION USER AN AMPER
BNE   *+1Ø                NO-SKIP NEXT INST.
MVC   SPLDTUID,BLANKS     CLEAR IT.
CLI   PUNDPRGN,X'Ø1'      IS PROGRAMMER NAME LOW-VALUE.
BL    *+1Ø                YES-SKIP NEXT INST.
MVC   SPLDPRGN,PUNDPRGN   REPLACE IT.
CLI   PUNDROOM,X'Ø1'      IS ROOM NUMBER LOW-VALUE.
BL    *+1Ø                YES-SKIP NEXT INST.
MVC   SPLDROOM,PUNDROOM   REPLACE IT.
CLI   PUNDDEPT,X'Ø1'      IS DEPARTMENT NUMBER LOW-VALUE.
BL    *+1Ø                YES-SKIP NEXT INST.
MVC   SPLDDEPT,PUNDDEPT   REPLACE IT.
```

```
         CLI   PUNDBLDG,X'Ø1'      IS BUILDING NUMBER LOW-VALUE.
         BL    *+1Ø                YES-SKIP NEXT INST.
         MVC   SPLDBLDG,PUNDBLDG   REPLACE IT.
         CLI   PUNONCPY,X'Ø1'      IS NUMBER OF COPIES LOW-VALUE.
         BL    POWPU3              YES-BRANCH TO POWPU3.
         LA    11,PUNONCPY         LOAD ADDRESS OF EBCDIC CHARACTERS TO
         BAL   1Ø,POWCTH           PERFORM POWCTH ROUTINE.
         MVC   SPLONCPY,HEXCHR     MVE CONVERTED NUMBER OF COPIES.
*
POWPU3   EQU   *
         CLI   PUNOFORM,X'Ø1'      IS FORM NUMBER LOW-VALUE.
         BL    *+1Ø                YES-SKIP NEXT INST.
         MVC   SPLOFORM(4),PUNOFORM REPLACE IT.
         CLI   PUNOFORM,C'&&'      IS FORM NUMBER AN AMPERSAND (&).
         BNE   *+1Ø                NO-SKIP NEXT INST.
         MVC   SPLOFORM(8),BLANKS  CLEAR IT.
         CLI   PUNONSEP,X'Ø1'      IS NUMBER OF SEPARATOR CARDS LOW VAL
         BL    POWPU5              YES-BRANCH TO POWPU5.
         LA    11,PUNONSEP         LOAD ADDRESS OF EBCDIC CHARACTERS TO
         BAL   1Ø,POWCTH           PERFORM POWCTH ROUTINE.
         MVC   SPLONSEP,HEXCHR     MVE CONVERTED NUMBER OF SEPARATOR CA
*
POWPU5   EQU   *
         CLI   PUNGUS,X'41'        IS ORIGINATING USER ID HEXZERO/BLANK
         BL    POWNXT              YES-BRANCH TO POWNXT.
         MVC   SPLGUS,PUNGUS       REPLACE IT.
*        CLI   PUNGUS,C'&&'        IS ORIGINATING USER ID AN AMPERSAND
*        BNE   POWNXT              NO-BRANCH TO POWNXT.
*        MVC   SPLGUS,BLANKS       CLEAR IT.
         B     POWNXT              BRANCH TO POWNXT.
*
POWLS1   EQU   *
*        XC    POWPWD,POWPWD       CLEAR PASSWORD.
         MVC   POWPWD,=8C' '       CLEAR PASSWORD.
         PWRSPL TYPE=UPD,SPL=OWNSPL,QUEUE=LST,REQ=PUT,
               JOBN=JOBNME,PWD=POWPWD,USERID=USERID
         MVC   SPLDLREC,H132       SET MAXIMUM RECORD LENGTH TO 132.
         MVI   SPLORCFM,SPLORMCC   INDICATE RECORD FORMAT IS MACH CODE.
         CLI   LSTJOB,X'Ø1'        IS JOB NAME LOW-VALUE.
         BL    *+1Ø                YES-SKIP NEXT INST.
         MVC   SPLGJB,LSTJOB       REPLACE IT.
         CLI   LSTGCL,X'Ø1'        IS OUTPUT CLASS LOW-VALUE.
         BL    *+1Ø                YES-SKIP NEXT INST.
         MVC   SPLGCL,LSTGCL       REPLACE IT.
         CLI   LSTGPW,X'Ø1'        IS PASSWORD LOW-VALUE.
         BL    *+1Ø                YES-SKIP NEXT INST.
         MVC   SPLGPW,LSTGPW       REPLACE IT.
         CLI   LSTGPW,C'&&'        IS PASSWORD AN AMPERSAND (&).
         BNE   *+1Ø                NO-SKIP NEXT INST.
```

```
         MVC   SPLGPW,BLANKS       CLEAR IT.
         CLI   LSTDDP,X'Ø1'        IS OUTPUT DISPOSITION LOW-VALUE.
         BL    *+1Ø                YES-SKIP NEXT INST.
         MVC   SPLDDP,LSTDDP       REPLACE IT.
         CLI   LSTDPR,X'Ø1'        IS PRIORITY LOW-VALUE.
         BL    *+1Ø                YES-SKIP NEXT INST.
         MVC   SPLDPR,LSTDPR       REPLACE IT.
         CLI   LSTDSID,X'Ø1'       IS OUTPUT SYSTEM IDENTIFIER LOW VALU
         BL    *+1Ø                YES-SKIP NEXT INST.
         MVC   SPLDSID,LSTDSID     REPLACE IT.
         CLI   LSTDUI,X'Ø1'        IS USER INFORMATION LOW-VALUE.
         BL    *+1Ø                YES-SKIP NEXT INST.
         MVC   SPLDUI,LSTDUI       REPLACE IT.
         CLI   LSTDUI,C'&&'        IS USER INFORMATION AN AMPERSAND (&)
         BNE   *+1Ø                NO-SKIP NEXT INST.
         MVC   SPLDUI,BLANKS       CLEAR IT.
         MVC   SPLDTNN,BLANKS
         CLI   LSTDTNN,X'Ø1'       IS NAME OF DESTINATION NODE LOW VALU
         BL    *+1Ø                YES-SKIP NEXT INST.
         MVC   SPLDTNN,LSTDTNN     REPLACE IT.
         CLI   LSTDTNN,C'&&'       IS NAME OF DESTINATION NODE AN AMPER
         BNE   *+1Ø                NO-SKIP NEXT INST.
         MVC   SPLDTNN,BLANKS      CLEAR IT.
         MVC   SPLDTUID,BLANKS
         CLI   LSTDTUID,X'Ø1'      IS NAME OF DESTINATION USER LOW VALU
         BL    *+1Ø                YES-SKIP NEXT INST.
         MVC   SPLDTUID,LSTDTUID   REPLACE IT.
         CLI   LSTDTUID,C'&&'      IS NAME OF DESTINATION USER AN AMPER
         BNE   *+1Ø                NO-SKIP NEXT INST.
         MVC   SPLDTUID,BLANKS     CLEAR IT.
         CLI   LSTDPRGN,X'Ø1'      IS PROGRAMMER NAME LOW-VALUE.
         BL    *+1Ø                YES-SKIP NEXT INST.
         MVC   SPLDPRGN,LSTDPRGN   REPLACE IT.
         CLI   LSTDROOM,X'Ø1'      IS ROOM NUMBER LOW-VALUE.
         BL    *+1Ø                YES-SKIP NEXT INST.
         MVC   SPLDROOM,LSTDROOM   REPLACE IT.
         CLI   LSTDDEPT,X'Ø1'      IS DEPARTMENT NUMBER LOW-VALUE.
         BL    *+1Ø                YES-SKIP NEXT INST.
         MVC   SPLDDEPT,LSTDDEPT   REPLACE IT.
         CLI   LSTDBLDG,X'Ø1'      IS BUILDING NUMBER LOW-VALUE.
         BL    *+1Ø                YES-SKIP NEXT INST.
         MVC   SPLDBLDG,LSTDBLDG   REPLACE IT.
         CLI   LSTONCPY,X'Ø1'      IS NUMBER OF COPIES LOW-VALUE.
         BL    POWLS3              YES-BRANCH TO POWLS3.
         LA    11,LSTONCPY         LOAD ADDRESS OF EBCDIC CHARACTERS TO
         BAL   1Ø,POWCTH           PERFORM POWCTH ROUTINE.
         MVC   SPLONCPY,HEXCHR     MVE CONVERTED NUMBER OF COPIES.
*
POWLS3   EQU   *
```

```
         CLI   LSTOFORM,X'Ø1'      IS FORM NUMBER LOW-VALUE.
         BL    *+1Ø                YES-SKIP NEXT INST.
         MVC   SPLOFORM(4),LSTOFORM REPLACE IT.
         CLI   LSTOFORM,C'&&'      IS FORM NUMBER AN AMPERSAND (&).
         BNE   *+1Ø                NO-SKIP NEXT INST.
         MVC   SPLOFORM(8),BLANKS  CLEAR IT.
         CLI   LSTOEWTR,X'Ø1'      IS OUTPUT EXTERNAL WRITER LOW VALUE.
         BL    *+1Ø                YES-SKIP NEXT INST.
         MVC   SPLOEWTR,LSTOEWTR   REPLACE IT.
         CLI   LSTOFCB,X'Ø1'       IS FCB IMAGE PHASE LOW-VALUE.
         BL    *+1Ø                YES-SKIP NEXT INST.
         MVC   SPLOFCB,LSTOFCB     REPLACE IT.
         CLI   LSTOFCB,C'&&'       IS FCB IMAGE PHASE AN AMPERSAND (&).
         BNE   *+1Ø                NO-SKIP NEXT INST.
         MVC   SPLOFCB,BLANKS      CLEAR IT.
         CLI   LSTOUCB,X'Ø1'       IS UCB IMAGE PHASE LOW-VALUE.
         BL    *+1Ø                YES-SKIP NEXT INST.
         MVC   SPLOUCB,LSTOUCB     REPLACE IT.
         CLI   LSTOUCB,C'&&'       IS UCB IMAGE PHASE AN AMPERSAND (&).
         BNE   *+1Ø                NO-SKIP NEXT INST.
         MVC   SPLOUCB,BLANKS      CLEAR IT.
         CLI   LSTOUCBO,C'F'       IS UCB OPTION FOR FOLD.
         BNE   *+12                NO-SKIP NEXT TWO (2) INST.
         MVI   SPLOUCBO,C' '       INDICATE UCB OPTION IS FOR FOLD.
         B     POWLS5              BRANCH TO POWLS5.
         CLI   LSTOUCBO,C'C'       IS UCB OPTION FOR CHECK.
         BNE   *+12                NO-SKIP NEXT TWO (2) INST.
         MVI   SPLOUCBO,X'8Ø'      INDICATE UCB OPTION IS FOR CHECK.
         B     POWLS5              BRANCH TO POWLS5.
         CLI   LSTOUCBO,C'B'       IS UCB OPTION FOR BOTH.
         BNE   POWLS5              NO-BRANCH TO POWLS5.
         MVI   SPLOUCBO,X'CØ'      INDICATE UCB OPTION IS FOR BOTH.
*
POWLS5   EQU   *
         CLI   LSTONSEP,X'Ø1'      IS NUMBER OF SEPARATOR PAGES LOW VAL
         BL    POWLS7              YES-BRANCH TO POWLS7.
         LA    11,LSTONSEP         LOAD ADDRESS OF EBCDIC CHARACTERS TO
         BAL   1Ø,POWCTH           PERFORM POWCTH ROUTINE.
         MVC   SPLONSEP,HEXCHR     MVE CONVERTED NUMBER OF SEPARATOR PA
*
POWLS7   EQU   *
         CLI   LSTCCODE,C'A'       IS COMMAND CODE FOR ASA.
         BNE   POWLS9              NO-BRANCH TO POWLS9.
         MVI   SPLORCFM,SPLORASA   INDICATE RECORD FORMAT IS ASA CODE.
         CLI   LSTCVASA,C'Y'       ARE WE CONVERTING ASA TO MACHINE.
         BNE   POWLS9              NO-BRANCH TO POWLS9.
         MVI   SPLGOPT2,SPLGO2AC   INDICATE CONVERT ASA TO MACHINE.
*
```

```
POWLS9   EQU   *
         CLI   LSTGUS,X'41'        IS ORIGINATING USER ID HEXZERO/BLANK
         BL    POWNXT              YES-BRANCH TO POWNXT.
         MVC   SPLGUS,LSTGUS       REPLACE IT.
*        CLI   LSTGUS,C'&&'        IS ORIGINATING USER ID AN AMPERSAND
*        BNE   POWNXT              NO-BRANCH TO POWNXT.
*        MVC   SPLGUS,BLANKS       CLEAR IT.
*
POWNXT   EQU   *
         MVI   PXUBTYP,PXUBTSPL    INDICATE BUFFER TYPE IS FOR SPL.
         MVI   PXUACT1,Ø           CLEAR ACTION BYTE.
         MVI   PXUSIGNL,Ø          ...
         STCM  6,7,IJBXADR         STORE SPL ADDRESS INTO XPCCB.
         LA    1Ø,SPLGLEN          LOAD LENGTH OF SPL TO REG 1Ø.
         ST    1Ø,IJBXBLN          STORE IT INTO XPCCB.
         BAL   1Ø,POWSND           PERFORM POWSND ROUTINE.
         CLI   PXPRETCD,PXPRCOK    WAS VSE/POWER RETURN CODE ZERO.
         BE    *+16                YES-SKIP NEXT THREE (3) INST.
         MVI   POWCDE,C'A'         INDICATE REQUEST FAILURE.
         BAL   1Ø,POWRCF           PERFORM POWRCF ROUTINE.
         B     POWDIS              BRANCH TO POWDIS.
         CLI   PXPFBKCD,PXPØØOK    WAS VSE/POWER FEEDBACK CODE ZERO.
         BE    POWBLD              YES-BRANCH TO POWBLD.
         MVI   POWCDE,C'A'         INDICATE REQUEST FAILURE.
         BAL   1Ø,POWRCF           PERFORM POWRCF ROUTINE.
         B     POWDIS              BRANCH TO POWDIS.
*
POWBLD   EQU   *
         MVI   PXUBTYP,PXUBTNDB    INDICATE BUFFER TYPE IS FOR NORMAL D
         MVI   PXUACT1,Ø           CLEAR ACTION BYTE.
         L     11,ASENDBUF         LOAD ADDRESS OF SENDBUF TO REG 11.
         STCM  11,7,IJBXADR        STORE IT INTO XPCCB.
         MVI   POWSSW,C'1'         INDICATE WE'RE IN PROCESS OF SPOOLING
*
POWCLS   EQU   *
         BAL   14,POWCSB           PERFORM POWCSB ROUTINE.
*
POWFIL   EQU   *
         L     11,ADDRESS          LOAD ADDRESS TO REG 11.
         CLC   POWREC(L'POWCTLR),POWCTLR IS THIS A CTL RECORD.
         BE    POWEOD              YES-BRANCH TO POWEOD.
         CLC   POWREC(L'POWEODR),POWEODR IS THIS A EOD RECORD.
         BE    POWEOD              YES-BRANCH TO POWEOD.
         C     11,=A(SENDEND-L'POWEOJR-141) IS BUFFER FULL.
         BH    POWFUL              YES-BRANCH TO POWFUL.
         CLI   POWRSW,C'1'         ARE WE SPOOLING TO RDR QUEUE.
         BNE   POWFI6              NO-BRANCH TO POWFI6.
         CLC   POWREC+1(4),POWJOB  IS THIS A VSE/POWER JECL STATEMENT
         BNE   POWFI6              NO-BRANCH TO POWFI6.
```

```
         MVC   OPTREC,OPTSAV
         CLC   RDRDTNN,BLANKS      IS RDR DESTINATION NODE BLANK.
         BH    POWFIØ              NO-BRANCH TO POWFIØ.
         CLC   RDRGPW,BLANKS       IS RDR PASSWORD BLANK.
         BH    POWFIØ              NO-BRANCH TO POWFIØ.
         CLI   OPTLDE,C'Y'         ARE WE TO INSERT LDEST= OPERAND.
         BE    POWFIØ              YES-BRANCH TO POWFIØ.
         CLI   OPTNTF,C'Y'         ARE WE TO INSERT NTFY=YES OPERAND.
         BNE   POWFI6              NO-BRANCH TO POWFI6.
*
POWFIØ   EQU   *
         LA    14,POWREC+1+3       LOAD FRONT END OF POWREC TO REG 14.
         LA    15,POWREC+71        LOAD BACK END OF POWREC TO REG 15.
*
POWFI1   EQU   *
         CLC   Ø(5,14),POWJOB+4    IS JECL STATEMENT A ' JOB ' STATEMEN
         BE    POWFI2              YES-BRANCH TO POWFI2.
         LA    14,1(14)            INCREMENT REG 14 TO NEXT POSITION.
         CR    14,15               ARE WE DONE.
         BL    POWFI1              NO-BRANCH TO POWFI1.
         B     POWFI6              BRANCH TO POWFI6. (NOT JOB
                                   STATEMENT
*
POWFI2   EQU   *
         LA    14,6(14)            INCREMENT REG 14 TO NEXT POSITION.
         MVI   POWJSW,C'1'         INDICATE PROCESSING JOB RECORD.
         MVC   POWCCH,POWREC+72    SVE POSSIBLE CONTINUED CHARACTER.
*
POWFI2A  EQU   *
         CLI   Ø(14),C' '          IS THIS POSITION BLANK.
         BE    POWFI2G             YES-BRANCH TO POWFI2G.
         CLC   =C',USER='',Ø(14)   IS THIS ,USER=' OPERAND.
         BNE   POWFI2AK            NO-BRANCH TO POWFI2AK.
         LA    14,7(14)            INCREMENT REG 14 PAST ,USER='.
*
POWFI2AC EQU   *
         CLI   Ø(14),X'7D'         IS THIS THE ENDING QUOTE.
         BE    POWFI2AK            YES-BRANCH TO POWFI2AK.
         LA    14,1(14)            INCREMENT TO NEXT POSITION.
         CR    14,15               ARE WE DONE.
         BL    POWFI2AC            NO-BRANCH TO POWFI2AC.
*
POWFI2AK EQU   *
*
```

The following commented instructions determine whether certain operands are already present in a VSE/POWER job statement.

```
*
*        CLI   OPTLDE,C'N'          DO WE INSERT LDEST= OPERAND.
*        BE    POWFI2AL             NO-BRANCH TO POWFI2AL.
*        CLC   Ø(6,14),POWBTHR+5    IS LDEST= ALREADY SPECIFIED.
*        BNE   POWFI2AL             NO-BRANCH TO POWFI2AL.
*        MVI   OPTLDE,C'N'          INDICATE NO INSERT LDEST=*
                                    OPERAND.
**
*OWFI2AL EQU   *
*        CLI   OPTNTF,C'N'          DO WE INSERT NTFY=YES OPERAND.
*        BE    POWFI2AM             NO-BRANCH TO POWFI2AM.
*        CLC   Ø(5,14),POWBTHR+13   IS NTFY= ALREADY SPECIFIED.
*        BNE   POWFI2AL             NO-BRANCH TO POWFI2AL.
*        MVI   OPTNTF,C'N'          INDICATE NO INSERT NTFY=YES
                                    OPERAND.
**
*OWFI2AM EQU   *
*        CLI   RDRGPW,X'41'         DO WE INSERT PWD OPERAND.
*        BL    POWFI2AO             NO-BRANCH TO POWFI2AO.
*        CLC   =C'PWD=',Ø(14)       IS PWD= ALREADY SPECIFIED.
*        BNE   POWFI2AO             NO-BRANCH TO POWFI2AO.
*        MVC   RDRGPW,BLANKS        CLEAR RDR PWD.
**
*OWFI2AO EQU   *
*        CLI   RDRDTNN,X'41'        DO WE INSERT XDEST OPERAND.
*        BL    POWFI2AP             NO-BRANCH TO POWFI2AP.
*        CLC   ='XDEST',Ø(14)       IS XDEST= ALREADY SPECIFIED.
*        BNE   POWFI2AP             NO-BRANCH TO POWFI2AP.
*        MVC   RDRDTNN,BLANKS       CLEAR RDR DESTINATION NODE.
*
POWFI2AP EQU   *
         CLI   RDRGPW,X'4Ø'         DO WE INSERT PWD OPERAND.
         BH    POWFI2C              YES-BRANCH TO POWFI2C.
         CLI   RDRDTNN,X'4Ø'        DO WE INSERT XDEST OPERAND.
         BH    POWFI2C              YES-BRANCH TO POWFI2C.
         CLI   OPTLDE,C'Y'          ARE WE TO INSERT LDEST= OPERAND.
         BE    POWFI2C              YES-BRANCH TO POWFI2C.
         CLI   OPTNTF,C'Y'          ARE WE TO INSERT NTFY=YES OPERAND.
         BNE   POWFI6               NO-BRANCH TO POWFI6.
*
POWFI2C  EQU   *
         LA    14,1(14)             INCREMENT REG 14 TO NEXT POSITION.
         CR    14,15                ARE WE DONE.
         BL    POWFI2A              NO-BRANCH TO POWFI2A.
         MVI   POWJSW,C'Ø'          INDICATE NOT PROCESSING JOB RECORD.
         B     POWFI6               BRANCH TO POWFI6.
*
POWFI2G  EQU   *
         BCTR  14,Ø                 BACK UP ONE (1) POSITION.
```

```
         CLI   Ø(14),C','          IS THIS POSITION A COMMA (,).
         BE    POWFI6              YES-BRANCH TO POWFI6.
         LA    14,1(14)            INCREMENT REG 14 TO NEXT POSITION.
         MVI   Ø(14),C','          MVE COMMA (,).
         LA    14,1(14)            INCREMENT REG 14 TO NEXT POSITION.
         CR    14,15               ARE WE DONE.
         BNH   POWFI5              NO-BRANCH TO POWFI5.
         MVI   POWJSW,C'Ø'         INDICATE NOT PROCESSING JOB RECORD.
         B     POWFI6              BRANCH TO POWFI6.
*
POWFI5   EQU   *
         MVI   Ø(14),C' '          CLEAR POSITION AFTER COMMA (,).
         MVI   POWREC+72,C'X'      MVE CONTINUED CHARACTER.
*
POWFI6   EQU   *
         AP    RECCNT,=P'1'        ADD ONE TO RECORD COUNT.
         MVC   POWPRTR,POWREC      MVE RECORD TO PRINT RECORD.
         BAL   1Ø,POWPRT           PERFORM POWPRT ROUTINE.
         USING RECPRFIX,11
         XC    Ø(RECPRFXL,11),Ø(11) CLEAR RECORD PREFIX.
         MVI   RECTYPE,RECTNORM    INDICATE NORMAL RECORD.
         CLI   POWREC,C'6'         ARE WE SPOOLING TO PUN QUEUE.
         BNE   *+8                 NO-SKIP NEXT INST.
         MVI   RECCCODE,X'Ø1'      INDICATE COMMAND CODE.
         CLI   POWREC,C'8'         ARE WE SPOOLING TO LST QUEUE.
         BNE   *+1Ø                NO-SKIP NEXT INST.
         MVC   RECCCODE,POWREC+1   MVE COMMAND CODE.
         LA    1Ø,L'POWREC-1       LOAD RECORD LENGTH TO REG 1Ø. (8Ø).
         CLI   POWREC,C'4'         ARE WE SPOOLING 128 BYTE RECORDS.
         BNE   *+8                 NO-SKIP NEXT INST.
         LA    1Ø,L'POWREC-1+48    LOAD RECORD LENGTH TO REG 1Ø. (128).
         CLI   POWREC,C'8'         ARE WE SPOOLING 133 BYTE RECORDS.
         BNE   *+8                 NO-SKIP NEXT INST.
         LA    1Ø,L'POWREC-1+52    LOAD RECORD LENGTH TO REG 1Ø. (132).
         STH   1Ø,RECLNGTH         STORE IT INTO RECORD LENGTH.
         DROP  11
         CLI   POWREC,C'4'         ARE WE SPOOLING 128 BYTE RECORDS.
         BNE   *+18                NO-SKIP NEXT THREE (3) INST.
         MVC   8(L'POWREC-1+48,11),POWREC+1 MVE PASSED RECORD TO BUFFER
         LA    11,L'POWREC-1+48+8(11) INCREMENT TO NEXT BUFFER POSITION
         B     *+36                SKIP NEXT SEVEN (7) INST.
         CLI   POWREC,C'8'         ARE WE SPOOLING 133 BYTE RECORDS.
         BNE   *+18                NO-SKIP NEXT THREE (3) INST.
         MVC   8(L'POWREC-1+52,11),POWREC+2 MVE PASSED RECORD TO BUFFER
         LA    11,L'POWREC-1+52+8(11) INCREMENT TO NEXT BUFFER POSITION
         B     *+14                SKIP NEXT TWO (2) INST.
         MVC   8(L'POWREC-1,11),POWREC+1 MVE PASSED RECORD TO BUFFER AR
         LA    11,L'POWREC-1+8(11) INCREMENT TO NEXT BUFFER POSITION.
         MVC   POWRECS,POWREC      SVE FIRST TEN BYTES OF RECORD.
```

```
         CLI   POWRSW,C'1'         ARE WE SPOOLING TO RDR QUEUE.
         BNE   POWFI9              NO-BRANCH TO POWFI9.
         CLI   POWJSW,C'Ø'         ARE WE PROCESSING JOB RECORD.
         MVI   POWJSW,C'Ø'         INDICATE NOT PROCESSING JOB RECORD.
         BE    POWFI9              NO-BRANCH TO POWFI9.
         CLI   OPTLDE,C'N'         DO WE INSERT LDEST=* OPERAND. .
         BE    POWFI7              NO-BRANCH TO POWFI7.
         ST    11,ADDRESS          STORE IT IN ADDRESS.
         USING RECPRFIX,11
         XC    Ø(RECPRFXL,11),Ø(11) CLEAR RECORD PREFIX.
         MVI   RECTYPE,RECTNORM    INDICATE NORMAL RECORD.
         LA    1Ø,8Ø               LOAD RECORD LENGTH TO REG 1Ø.
         STH   1Ø,RECLNGTH         STORE IT INTO RECORD LENGTH.
         DROP  11
         L     11,ADDRESS          LOAD ADDRESS TO REG 11.
         MVI   POWPRTR,C' '        CLEAR PRINT RECORD.
         MVC   POWPRTR+1(L'POWPRTR-1),POWPRTR ...
         LA    1Ø,POWPRTR          LOAD ADDRESS OF PRINT RECORD TO REG
         MVC   Ø(12,1Ø),POWBTHR    SET JECL PREFIX. ('* $$ ').
         LA    1Ø,12(1Ø)           INCREMENT TO NEXT POSITION.
         CLI   POWCCH,C' '         IS STATEMENT CONTINUED.
         BH    POWFI6A             YES-BRANCH TO POWFI6A.
         CLI   OPTNTF,C'Y'         DO WE INSERT NTFY=YES OPERAND.
         BE    POWFI6A             YES-BRANCH TO POWFI6A.
         CLI   RDRGPW,X'4Ø'        DO WE INSERT PWD OPERAND.
         BH    POWFI6A             YES-BRANCH TO POWFI6A.
         CLI   RDRDTNN,X'41'       DO WE INSERT XDEST OPERAND.
         BL    POWFI6C             NO-BRANCH TO POWFI6C.
*
POWFI6A  EQU   *
         MVI   Ø(1Ø),C','          MVE COMMA (,).
         MVI   POWPRTR+71,C'X'     MVE CONTINUED CHARACTER.
*
POWFI6C  EQU   *
         MVC   8(8Ø,11),POWPRTR MVE PRINT RECORD TO BUFFER.
         MVI   POWPRTR,C' '        CLEAR PRINT RECORD.
         MVC   POWPRTR+1(L'POWPRTR-1),POWPRTR ...
         MVI   POWPRTR,C'3'
         MVC   POWPRTR+1(L'POWPRTR-1),8(11)
         LA    11,8Ø+8(11)         INCREMENT TO NEXT BUFFER POSITION.
         BAL   1Ø,POWPRT           PERFORM POWPRT ROUTINE.
         AP    RECCNT,=P'1'        ADD ONE TO RECORD COUNT.
*
POWFI7   EQU   *
         CLI   OPTNTF,C'N'         DO WE INSERT NTFY=YES OPERAND.
         BE    POWFI8              NO-BRANCH TO POWFI8.
         ST    11,ADDRESS          STORE IT IN ADDRESS.
         USING RECPRFIX,11
         XC    Ø(RECPRFXL,11),Ø(11) CLEAR RECORD PREFIX.
```

```
         MVI   RECTYPE,RECTNORM    INDICATE NORMAL RECORD.
         LA    1Ø,8Ø               LOAD RECORD LENGTH TO REG 1Ø.
         STH   1Ø,RECLNGTH         STORE IT INTO RECORD LENGTH.
         DROP  11
         L     11,ADDRESS          LOAD ADDRESS TO REG 11.
         MVI   POWPRTR,C' '        CLEAR PRINT RECORD.
         MVC   POWPRTR+1(L'POWPRTR-1),POWPRTR ...
         LA    1Ø,POWPRTR          LOAD ADDRESS OF PRINT RECORD TO REG
         MVC   Ø(5,1Ø),POWBTHR     SET JECL PREFIX. ('* $$ ').
         MVC   5(8,1Ø),POWBTHR+13  MVE 'NTFY=YES'.
         LA    1Ø,13(1Ø)           INCREMENT TO NEXT POSITION.
         CLI   POWCCH,C' '         IS STATEMENT CONTINUED.
         BH    POWFI7A             YES-BRANCH TO POWFI7A.
         CLI   RDRGPW,X'4Ø'        DO WE INSERT PWD OPERAND.
         BH    POWFI7A             YES-BRANCH TO POWFI7A.
         CLI   RDRDTNN,X'41'       DO WE INSERT XDEST OPERAND.
         BL    POWFI7C             NO-BRANCH TO POWFI7C.
*
POWFI7A  EQU   *
         MVI   Ø(1Ø),C','          MVE COMMA (,).
         MVI   POWPRTR+71,C'X'     MVE CONTINUED CHARACTER.
*
POWFI7C  EQU   *
         MVC   8(8Ø,11),POWPRTR MVE PRINT RECORD TO BUFFER.
         MVI   POWPRTR,C' '        CLEAR PRINT RECORD.
         MVC   POWPRTR+1(L'POWPRTR-1),POWPRTR ...
         MVI   POWPRTR,C'3'
         MVC   POWPRTR+1(L'POWPRTR-1),8(11)
         LA    11,8Ø+8(11)         INCREMENT TO NEXT BUFFER POSITION.
         BAL   1Ø,POWPRT           PERFORM POWPRT ROUTINE.
         AP    RECCNT,=P'1'        ADD ONE TO RECORD COUNT.
*
POWFI8   EQU   *
         CLI   RDRGPW,X'41'        DO WE INSERT PWD OPERAND.
         BL    POWFI8J             NO-BRANCH TO POWFI8J.
         ST    11,ADDRESS          STORE IT IN ADDRESS.
         USING RECPRFIX,11
         XC    Ø(RECPRFXL,11),Ø(11) CLEAR RECORD PREFIX.
         MVI   RECTYPE,RECTNORM    INDICATE NORMAL RECORD.
         LA    1Ø,8Ø               LOAD RECORD LENGTH TO REG 1Ø.
         STH   1Ø,RECLNGTH         STORE IT INTO RECORD LENGTH.
         DROP  11
         L     11,ADDRESS          LOAD ADDRESS TO REG 11.
         MVI   POWPRTR,C' '        CLEAR PRINT RECORD.
         MVC   POWPRTR+1(L'POWPRTR-1),POWPRTR ...
         LA    1Ø,POWPRTR          LOAD ADDRESS OF PRINT RECORD TO REG
         MVC   71(1,1Ø),POWCCH     MVE POSSIBLE CONTINUED CHARACTER.
         MVC   Ø(5,1Ø),POWBTHR     SET JECL PREFIX. ('* $$ ').
         MVC   5(4,1Ø),=C'PWD='    MVE 'PWD='.
```

```
         MVC   9(8,1Ø),RDRGPW      MVE PWD.
         LA    1Ø,9(1Ø)            INCREMENT TO NEXT POSITION.
         CLI   POWCCH,C' '         IS STATEMENT CONTINUED.
         BH    POWFI8A             YES-BRANCH TO POWFI8A.
         CLI   RDRDTNN,X'41'       DO WE INSERT XDEST OPERAND.
         BL    POWFI8G             NO-BRANCH TO POWFI8G.
*
POWFI8A  EQU   *
         CLI   Ø(1Ø),C' '          IS THIS POSITION BLANK.
         BE    POWFI8B             YES-BRANCH TO POWFI8B.
         LA    1Ø,1(1Ø)            INCREMENT TO NEXT POSITION.
         B     POWFI8A             BRANCH TO POWFI8A.
*
POWFI8B  EQU   *
         MVI   Ø(1Ø),C','          MVE COMMA (,).
         MVI   POWPRTR+71,C'X'     MVE CONTINUED CHARACTER.
*
POWFI8G  EQU   *
         MVC   8(8Ø,11),POWPRTR MVE PRINT RECORD TO BUFFER.
         MVI   POWPRTR,C' '        CLEAR PRINT RECORD.
         MVC   POWPRTR+1(L'POWPRTR-1),POWPRTR ...
         MVI   POWPRTR,C'3'
         MVC   POWPRTR+1(L'POWPRTR-1),8(11)
         LA    11,8Ø+8(11)         INCREMENT TO NEXT BUFFER POSITION.
         BAL   1Ø,POWPRT           PERFORM POWPRT ROUTINE.
         AP    RECCNT,=P'1'        ADD ONE TO RECORD COUNT.
*
POWFI8J  EQU   *
         CLI   RDRDTNN,X'41'       DO WE INSERT XDEST OPERAND.
         BL    POWFI9              NO-BRANCH TO POWFI9.
         ST    11,ADDRESS          STORE IT IN ADDRESS.
         USING RECPRFIX,11
         XC    Ø(RECPRFXL,11),Ø(11) CLEAR RECORD PREFIX.
         MVI   RECTYPE,RECTNORM    INDICATE NORMAL RECORD.
         LA    1Ø,8Ø               LOAD RECORD LENGTH TO REG 1Ø.
         STH   1Ø,RECLNGTH         STORE IT INTO RECORD LENGTH.
         DROP  11
         L     11,ADDRESS          LOAD ADDRESS TO REG 11.
         MVI   POWPRTR,C' '        CLEAR PRINT RECORD.
         MVC   POWPRTR+1(L'POWPRTR-1),POWPRTR ...
         LA    1Ø,POWPRTR          LOAD ADDRESS OF PRINT RECORD TO
                                   REG
         MVC   71(1,1Ø),POWCCH     MVE POSSIBLE CONTINUED CHARACTER.
         MVC   Ø(5,1Ø),POWBTHR     SET JECL PREFIX. ('* $$ ').
         MVC   5(6,1Ø),=C'XDEST='  MVE 'XDEST='.
         MVC   11(8,1Ø),RDRDTNN    MVE XDEST NODE.
         LA    1Ø,11(1Ø)           INCREMENT TO NEXT POSITION.
         CLI   POWCCH,C' '         IS STATEMENT CONTINUED.
         BE    POWFI8S             NO-BRANCH TO POWFI8S.
```

```
*
POWFI8M  EQU   *
         CLI   Ø(1Ø),C' '          IS THIS POSITION BLANK.
         BE    POWFI8P             YES-BRANCH TO POWFI8P.
         LA    1Ø,1(1Ø)            INCREMENT TO NEXT POSITION.
         B     POWFI8M             BRANCH TO POWFI8M.
*
POWFI8P  EQU   *
         MVI   Ø(1Ø),C','          MVE COMMA (,).
*
POWFI8S  EQU   *
         MVC   8(8Ø,11),POWPRTR MVE PRINT RECORD TO BUFFER.
         MVI   POWPRTR,C' '        CLEAR PRINT RECORD.
         MVC   POWPRTR+1(L'POWPRTR-1),POWPRTR ...
         MVI   POWPRTR,C'3'
         MVC   POWPRTR+1(L'POWPRTR-1),8(11)
         LA    11,8Ø+8(11)         INCREMENT TO NEXT BUFFER POSITION.
         BAL   1Ø,POWPRT           PERFORM POWPRT ROUTINE.
         AP    RECCNT,=P'1'        ADD ONE TO RECORD COUNT.
*
POWFI9   EQU   *
         MVI   POWCCH,C' '         CLEAR JOB CONTINUED CHARACTER.
         ST    11,ADDRESS          STORE IT IN ADDRESS.
         CLI   POWREC,C'9'         IS THIS THE LAST RECORD BEING
                                   PASSED
         BE    POWEOD3             YES-BRANCH TO POWEOD3.
         B     POWRTN              BRANCH TO POWRTN.
*
POWFUL   EQU   *
         L     1Ø,ASENDBUF         LOAD BEGIN ADDRESS OF BUFFER TO REG
         ST    11,SV11A
         SR    11,1Ø               CALCULATE BUFFER LENGTH.
         ST    11,IJBXBLN          STORE IT INTO XPCCB.
         ST    11,SV11B
         BAL   1Ø,POWSND           PERFORM POWSND ROUTINE.
         CLI   PXPRETCD,PXPRCOK    WAS VSE/POWER RETURN CODE ZERO.
         BE    *+16                YES-SKIP NEXT THREE (3) INST.
         MVI   POWCDE,C'B'         INDICATE REQUEST FAILURE.
         BAL   1Ø,POWRCF           PERFORM POWRCF ROUTINE.
         B     POWDIS              BRANCH TO POWDIS.
         CLI   PXPFBKCD,PXPØØOK    WAS VSE/POWER FEEDBACK CODE ZERO.
         BE    POWCLS              YES-BRANCH TO POWCLS.
         MVI   POWCDE,C'B'         INDICATE REQUEST FAILURE.
         BAL   1Ø,POWRCF           PERFORM POWRCF ROUTINE.
         B     POWDIS              BRANCH TO POWDIS.
*
POWEOD   EQU   *
         MVC   POWPRTR,Ø(7)        MVE RECORD TO PRINT RECORD.
         BAL   1Ø,POWPRT           PERFORM POWPRT ROUTINE.
```

```
*
POWEOD3  EQU   *
         CLI   POWRSW,C'Ø'          ARE WE SPOOLING TO RDR QUEUE.
         BE    POWBYP               NO-BRANCH TO POWBYP.
         CLC   POWRECS+1(9),POWEOJR WAS LAST RECORD A VSE/POWER EOJ STA
         BE    POWBYP               YES-BRANCH TO POWBYP.
         USING RECPRFIX,11
         XC    Ø(RECPRFXL,11),Ø(11) CLEAR RECORD PREFIX.
         MVI   RECTYPE,RECTNORM     INDICATE NORMAL RECORD.
         LA    1Ø,L'POWEOJR         LOAD RECORD LENGTH TO REG 1Ø.
         STH   1Ø,RECLNGTH          STORE IT INTO RECORD LENGTH.
         DROP  11
         MVC   8(L'POWEOJR,11),POWEOJR MVE VSE/POWER EOJ STATEMENT TO B
         LA    11,L'POWEOJR+8(11)   INCREMENT TO NEXT BUFFER POSITION.
*
POWBYP   EQU   *
         L     1Ø,ASENDBUF          LOAD BEGIN ADDRESS OF BUFFER.
         ST    11,SV11C
         SR    11,1Ø                CALCULATE BUFFER LENGTH.
         ST    11,IJBXBLN           STORE IT INTO XPCCB.
         ST    11,SV11D
         MVI   PXUACT1,PXUATEOD     INDICATE END OF DATA.
         BAL   1Ø,POWSND            PERFORM POWSND ROUTINE.
         CLI   PXPRETCD,PXPRCOK     WAS VSE/POWER RETURN CODE ZERO.
         BE    *+16                 YES-SKIP NEXT THREE (3) INST.
         MVI   POWCDE,C'C'          INDICATE REQUEST FAILURE.
         BAL   1Ø,POWRCF            PERFORM POWRCF ROUTINE.
         B     POWDIS               BRANCH TO POWDIS.
         CLI   PXPFBKCD,PXPØØOK     WAS VSE/POWER FEEDBACK CODE ZERO.
         BE    POWMSG               YES-BRANCH TO POWMSG.
         MVI   POWCDE,C'C'          INDICATE REQUEST FAILURE.
         BAL   1Ø,POWRCF            PERFORM POWRCF ROUTINE.
         B     POWDIS               BRANCH TO POWDIS.
*
POWMSG   EQU   *
         LA    11,REPLBUF           LOAD ADDRESS OF REPLBUF TO REG 11.
         USING OWNSPLDS,11
         LA    1Ø,SPLGLEN           LOAD LENGTH OF SPL TO REG 1Ø.
         STC   1Ø,*+5               INSERT IT INTO FOLLOWING MVC INST.
         MVC   POWREP(1),OWNSPLDS   SVE ENTIRE SPL.
         DROP  11
         USING OWNSPLDS,6
         TM    PXPINFO,PXPIMSG      ANY MESSAGES.
         BNO   POWDIS               NO-BRANCH TO POWDIS.
         XC    IJBXBLN,IJBXBLN      CLEAR BUFFER LENGTH.
         MVI   PXUBTYP,X'ØØ'        CLEAR BUFFER TYPE.
         MVI   PXUACT1,PXUATRMR     INDICATE RETURN MESSAGES.
         BAL   1Ø,POWSND            PERFORM POWSND ROUTINE.
         CLI   PXPRETCD,PXPRCOK     WAS VSE/POWER RETURN CODE ZERO.
```

```
         BE    *+12                YES-SKIP NEXT TWO (2) INST.
         MVI   POWCDE,C'9'         INDICATE REQUEST FAILURE.
         BAL   1Ø,POWRCF           PERFORM POWRCF ROUTINE.
         CLI   PXPFBKCD,PXPØØOK    WAS VSE/POWER FEEDBACK CODE ZERO.
         BE    POWDIS              YES-BRANCH TO POWDIS.
         MVI   POWCDE,C'9'         INDICATE REQUEST FAILURE.
         BAL   1Ø,POWRCF           PERFORM POWRCF ROUTINE.
*
POWDIS   EQU   *
         XPCC  XPCCB=(3),FUNC=DISCONN DISCONNECT LINK TO VSE/POWER.
         LTR   15,15               WAS DISCONNECT SUCCESSFUL.
         BZ    POWTRM              YES-BRANCH TO POWTRM.
         CLI   IJBXRETC,X'15'      IS RETURN CODE X'15'.
         BNE   POWDIS3             NO-BRANCH TO POWDIS3.
*
POWDIS3  EQU   *
         MVI   POWCDE,C'7'         INDICATE DISCONNECT FAILURE.
         BAL   1Ø,POWXRC           PERFORM POWXRC ROUTINE.
         B     POWFIN              BRANCH TO POWFIN.
*
POWTRM   EQU   *
         XPCC  XPCCB=(3),FUNC=TERMIN TERMINATE XPCC SUPPORT.
         LTR   15,15               WAS TERMINATION SUCCESSFUL.
         BZ    POWFIN              YES-BRANCH TO POWFIN.
         MVI   POWCDE,C'8'         INDICATE TERMINATION FAILURE.
         BAL   1Ø,POWXRC           PERFORM POWXRC ROUTINE.
*
POWFIN   EQU   *
         BAL   1Ø,POWRST           PERFORM POWRST ROUTINE.
*
POWRTN   EQU   *
         MVC   Ø(L'POWREC,7),POWREC
         CLI   POWREC,C'4'         WERE WE SPOOLING 128 BYTE RECORDS.
         BNE   *+1Ø                NO-SKIP NEXT INST.
         MVC   Ø(L'POWREC+48,7),POWREC
         CLI   POWREC,C'8'         WERE WE SPOOLING 133 BYTE RECORDS.
         BNE   POWEXT              NO-BRANCH TO POWEXT.
         MVC   Ø(L'POWREC+53,7),POWREC
*
POWEXT   EQU   *
         MVC   Ø(25Ø,8),POWCDE
         MVC   25Ø(255,8),POWCDE+25Ø
         MVC   POWPRC,POWLRC       MVE LAST RECORD TO PREVIOUS.
         MVC   POWLRC,POWREC       MVE CURRENT RECORD TO LAST.
         CLI   POWCDE,C'Ø'         WERE THERE ANY ERRORS.
         BE    *+22                NO-SKIP NEXT FOUR (4) INST.
         CLC   POWCDE,=C'19'       WAS ERROR FOR NO ROOM.
         BE    *+12                YES-SKIP NEXT TWO (2) INST.
         BAL   14,POWCSB           PERFORM POWCSB ROUTINE.
```

```
         BAL   1Ø,POWRST           PERFORM POWRST ROUTINE.
         L     13,SAVEAREA+4       LOAD RETURN ADDRESS TO REG 13.
         RETURN (14,12)            RETURN TO CALLER.
*
POWCSB   EQU   *
         L     1Ø,ASENDBUF         LOAD ADDRESS OF SENDBUF TO REG 1Ø.
         LA    11,L'SENDBUF        LOAD LENGTH OF SENDBUF TO REG 11.
         L     15,=X'ØØØØØØØØ'     SET PAD BYTE AND LENGTH FOR MVCL.
         MVCL  1Ø,14               CLEAR IT.
         L     11,ASENDBUF         LOAD ADDRESS OF SENDBUF TO REG 11.
         ST    11,ADDRESS          STORE IT IN ADDRESS.
         BR    14                  RETURN TO CALLER.
*
POWRST   EQU   *
         MVC   POWSSW(4),=C'ØØØØØ' CLEAR SWITCHES.
         XC    RDRREC,RDRREC       CLEAR RDRREC.
         XC    PUNREC,PUNREC       CLEAR PUNREC.
         XC    LSTREC,LSTREC       CLEAR LSTREC.
         BR    1Ø                  RETURN TO CALLER.
*
POWSND   EQU   *
         XPCC  XPCCB=(3),FUNC=SENDR SEND BUFFER TO VSE/POWER.
         LTR   15,15               WAS RETURN CODE ZERO.
         BZ    POWSND3             YES-BRANCH TO POWSND3.
         MVI   POWCDE,C'5'         INDICATE SENDR FAILURE.
         BAL   1Ø,POWXRC           PERFORM POWXRC ROUTINE.
         B     POWRTN              BRANCH TO POWRTN.
*
POWSND3  EQU   *
         CLI   POWCSW,C'1'         ARE WE RUNNING UNDER ICCF/CICS.
         BE    POWSNDC3            YES-BRANCH TO POWSNDC3.
         ST    1Ø,SV1ØB            SVE CONTENTS OF REG 1Ø.
         LA    1Ø,IJBXSECB         LOAD ADDRESS OF SEND COMPLETION ECB.
         WAIT  (1Ø)                WAIT FOR COMPLETION OF SENDR.
         L     1Ø,SV1ØB            RESTORE CONTENTS OF REG 1Ø.
*
POWSNDB3 EQU   *
         CLI   IJBXREAS,X'ØØ'      DID ANY CONNECTION ERROR OCCUR.
         BER   1Ø                  NO-RETURN TO CALLER.
         CLI   POWCSW,C'Ø'         ARE WE RUNNING UNDER ICCF/CICS.
         BE    POWSNDB5            NO-BRANCH TO POWSNDB5.
         AP    POWCNT,=P'1'        ADD ONE (1) TO WAIT COUNTER.
         CP    POWCNT,=P'15'       HAVE WE TRIED 15 TIMES.
         BL    POWSNDC5            NO-BRANCH TO POWSNDC5.
*
POWSNDB5 EQU   *
         MVI   POWCDE,C'6'         INDICATE SENDR CONNECTION FAILURE.
         BAL   1Ø,POWXRC           PERFORM POWXRC ROUTINE.
         B     POWRTN              BRANCH TO POWRTN.
```

```
*
POWSNDC3 EQU   *
         SP    POWCNT,POWCNT       CLEAR COUNTER.
*
POWSNDC5 EQU   *
         USING DFHCSADS,13
         L     13,ADDRCSA          ADDRESS CSA.
         L     12,CSACDTA          ADDRESS CURRENT DISPATCHED USER TCA.
         LA    11,IJBXSECB         LOAD ADDRESS OF SEND COMPLETION ECB
         ST    11,TCATCEA          STORE IT IN TCATCEA.
         DFHKC TYPE=WAIT,DCI=SINGLE
*        DFHIC TYPE=WAIT,INTRVAL=2
         DROP  13
         B     POWSNDB3            BRANCH TO POWSNDB3.
*
POWXRC   EQU   *              XPCC RETURN/REASON CODES CONVERSION ROUTI
         ST    3,SVØ3                SVE CONTENTS OF REG 3.
         ST    1Ø,SV1ØA              SVE CONTENTS OF REG 1Ø.
         ST    13,SV13               SVE CONTENTS OF REG 13.
         ST    15,SV15               SVE CONTENTS OF REG 15.
         MVC   SXPCCB,Ø(3)           SVE XPCCB.
         SR    13,13                 CLEAR REG 13.
         IC    13,IJBXRETC           INSERT XPCC RETURN CODE TO REG 13.
         BAL   1Ø,POWHTC             PERFORM POWHTC ROUTINE.
         MVC   POWCDE+1(L'RETCDE),RETCDE
         SR    13,13                 CLEAR REG 13.
         IC    13,IJBXREAS           INSERT XPCC REASON CODE TO REG 13.
         BAL   1Ø,POWHTC             PERFORM POWHTC ROUTINE.
         MVC   POWCDE+3(L'RETCDE),RETCDE
         L     1Ø,SV1ØA              RESTORE CONTENTS OF REG 1Ø.
         L     13,SV13               RESTORE CONTENTS OF REG 13.
         BR    1Ø                    RETURN TO CALLER.
*
POWRCF   EQU   *              VSE/POWER RETURN/FEEDBACK CODES
                              CONVERSIO
         ST    1Ø,SV1ØA              SVE CONTENTS OF REG 1Ø.
         ST    13,SV13               SVE CONTENTS OF REG 13.
         MVC   POWREP(25Ø),REPLBUF MVE REPLBUF.
         MVC   POWREP+25Ø(25Ø),REPLBUF+25Ø ...
         SR    13,13                 CLEAR REG 13.
         IC    13,PXPRETCD           INSERT VSE/POWER RETURN CODE TO REG
         BAL   1Ø,POWHTC             PERFORM POWHTC ROUTINE.
         MVC   POWCDE+1(L'RETCDE),RETCDE
         SR    13,13                 CLEAR REG 13.
         IC    13,PXPFBKCD           INSERT VSE/POWER FEEDBACK CODE TO RE
         BAL   1Ø,POWHTC             PERFORM POWHTC ROUTINE.
         MVC   POWCDE+3(L'RETCDE),RETCDE
         L     1Ø,SV1ØA              RESTORE CONTENTS OF REG 1Ø.
         L     13,SV13               RESTORE CONTENTS OF REG 13.
```

```
         BR    1Ø                  RETURN TO CALLER.
*
POWHTC   EQU   *              CONVERT ONE HEXADECIMAL CHARACTER TO TWO
         LA    11,RETCDE           LOAD ADDRESS OF OUTPUT TO REG 11.
         SLDL  12,28               SHIFT LEFT HALF.
         STC   12,Ø(11)            STORE IT.
         SRL   13,28               SHIFT RIGHT HALF.
         STC   13,1(11)            STORE IT.
         TR    Ø(L'RETCDE,11),TRTAB TRANSLATE BOTH.
         BR    1Ø                  RETURN TO CALLER.
*
POWCTH   EQU   *              CONVERT TWO EBCDIC CHARACTERS TO ONE HEXA
         NI    POWCTH2+1,X'ØF'     SET POWCTH2 NOT TO BRANCH TO POWCTH3
*
POWCTH1  EQU   *
         MVI   HOLDLIT,X'ØA'       ANTICIPATE AN A.
         CLI   Ø(11),C'A'          IS CHARACTER AN 'A'.
         BE    POWCTH2             YES-BRANCH TO POWCTH2.
         MVI   HOLDLIT,X'ØB'       ANTICIPATE A B.
         CLI   Ø(11),C'B'          IS CHARACTER A 'B'.
         BE    POWCTH2             YES-BRANCH TO POWCTH2.
         MVI   HOLDLIT,X'ØC'       ANTICIPATE A C.
         CLI   Ø(11),C'C'          IS CHARACTER A 'C'.
         BE    POWCTH2             YES-BRANCH TO POWCTH2.
         MVI   HOLDLIT,X'ØD'       ANTICIPATE A D.
         CLI   Ø(11),C'D'          IS CHARACTER A 'D'.
         BE    POWCTH2             YES-BRANCH TO POWCTH2.
         MVI   HOLDLIT,X'ØE'       ANTICIPATE A E.
         CLI   Ø(11),C'E'          IS CHARACTER A 'E'.
         BE    POWCTH2             YES-BRANCH TO POWCTH2.
         MVI   HOLDLIT,X'ØF'       ANTICIPATE A F.
         CLI   Ø(11),C'F'          IS CHARACTER A 'F'.
         BE    POWCTH2             YES-BRANCH TO POWCTH2.
         CLI   Ø(11),C'Ø'          IS CHARACTER LESS THAN ZERO (Ø).
         BL    POWCTHE             YES-BRANCH TO POWCTHE.
         CLI   Ø(11),C'9'          IS CHARACTER GREATER THAN NINE (9).
         BH    POWCTHE             YES-BRANCH TO POWCTHE.
         MVN   HOLDLIT,Ø(11)       MVE NUMERIC CHARACTER TO HOLDLIT.
*
POWCTH2  EQU   *
         BC    Ø,POWCTH3           IF SECOND PASS, BRANCH TO POWCTH3.
         PACK  HEXCHR,HOLDLIT      INVERT HOLDLIT INTO HEXCHR.
         LA    11,1(11)            SET REG 11 TO SECOND CHARACTER.
         OI    POWCTH2+1,X'FØ'     SET POWCTH2 TO BRANCH TO POWCTH3.
         B     POWCTH1             BRANCH TO POWCTH1.
*
POWCTH3  EQU   *
         MVN   HEXCHR,HOLDLIT      COMPLETE HEX LITERAL IN HEXCHR.
         BR    1Ø                  RETURN TO CALLER.
```

```
*
POWCTHE  EQU   *
         MVI   HEXCHR,X'ØØ'        INDICATE CHARACTERS INVALID.
         BR    1Ø                  RETURN TO CALLER.
*
POWPRT   EQU   *
         CLI   POWCDS+1,X'FE'      DO WE PRINT INCOMING RECORDS.
         BNER  1Ø                  NO-RETURN TO CALLER.
         MVC   LSTBUF+1(75),POWPRTR
         LA    1,LSTCCB            LOAD ADDRESS OF CCB.
         EXCP  (1)                 EXECUTE IT.
         WAIT  (1)                 WAIT FOR COMPLETION.
         MVC   LSTBUF+1(75),POWPRTR+75
         LA    1,LSTCCB            LOAD ADDRESS OF CCB.
         EXCP  (1)                 EXECUTE IT.
         WAIT  (1)                 WAIT FOR COMPLETION.
         MVC   LSTBUF,LSTBUF-1     CLEAR PRINTER WORK AREA.
         BR    1Ø                  RETURN TO CALLER.
*
DPPOWPS  DC    C'DPPOWP STORAGE HERE. ' INSERT EYE CATCHER.
MODE     DC    X'ØØ'
SIJBPIK  DS    CL2
JOBNME   DS    CL8
PGMNME   DS    CL8
RECCNT   DC    PL5'Ø'
POWCNT   DC    PL2'Ø'              ICCF/CICS LOOP COUNTER.
POWCDS   DS    CL5
POWPWD   DC    XL8'ØØ'             VSE/POWER PASSWORD.
SXPCCB   DS    CL96
TRTAB    DC    C'Ø123456789ABCDEF'
RETCDE   DC    C'ØØ'
HOLDLIT  DC    C' '
HEXCHR   DC    X'ØØ'
POWSSW   DC    C'Ø'                SPOOLING IN PROCESS SWITCH.
POWRSW   DC    C'Ø'                SPOOLING IN PROCESS FOR RDR SWITCH.
POWPSW   DC    C'Ø'                SPOOLING IN PROCESS FOR PUN SWITCH.
POWLSW   DC    C'Ø'                SPOOLING IN PROCESS FOR LST SWITCH.
POWJSW   DC    C'Ø'
POWCSW   DC    C'Ø'                RUNNING UNDER ICCF/CICS SWITCH.
POWREC   DC    CL81' '
         DC    CL48' '
         DC    CLØ5' '
         DC    CL17' '
POWCDE   DC    C'ØØØØØ'
POWREP   DS    CL5ØØ
OPID     DS    CL3
POWCCH   DC    C' '
POWEOJR  DS    ØCL17
         DC    C'* $$ EOJ '
```

```
USERID   DC    CL8'DPPOWP'
TERMID   DS    CL8
POWEODR  DC    C'9* $$ EOD  '
POWCTLR  DC    C'9* $$ CTL  '
POWTRMR  DC    C'9* $$ TRM  '
POWJOB   DC    C'* $$ JOB '
POWJOBR  DC    CL8Ø' '
POWBTHR  DC    C'* $$ LDEST=* NTFY=YES  '
         DC    C'POWRECS='
POWRECS  DS    CL1Ø
         DC    C'POWPRC='
POWPRC   DS    CL15Ø
         DC    C'POWLRC='
POWLRC   DS    CL15Ø
POWPRTR  DS    CL15Ø
BLANKS   DC    CL2Ø' '
*
         DC    C'OPTREC='
OPTREC   DS    ØCL11          ØØ1-Ø11 OPTION RECORD.
OPTRCDE  DC    C'1'                ØØ1-ØØ1 OPT RECORD CODE. (1).
OPTLDE   DC    C' '                ØØ2-ØØ2 OPT LDEST=*. (Y=YES, N=NO).
OPTNTF   DC    C' '                ØØ3-ØØ3 OPT NTFY=YES. (Y=YES, N=NO).
         DS    CL8                 Ø04-Ø11 FILLER. (RESERVED FOR
                                   FUTURE
*
OPTSAV   DS    CL11
*
RDRREC   DS    ØCL15Ø         ØØ1-15Ø RDR SETUP RECORD.
RDRRCDE  DS    C                   ØØ1-ØØ1 RDR RECORD CODE. (2).
RDRJOB   DC    CL8' '              ØØ2-ØØ9 RDR JOB NAME.
RDRGCL   DC    C' '                Ø1Ø-Ø1Ø RDR INPUT CLASS.
RDRGPW   DC    CL8' '              Ø11-Ø18 RDR PASSWORD.
RDRDDP   DC    C' '                Ø19-Ø19 RDR INPUT DISPOSITION.
RDRDPR   DC    C' '                Ø2Ø-Ø2Ø RDR INPUT PRIORITY.
RDRDSID  DC    C' '                Ø21-Ø21 RDR INPUT SYSTEM IDENTIFIER.
RDRDUI   DC    CL16' '             Ø22-Ø37 RDR USER INFORMATION.
RDRDTNN  DC    CL8' '              Ø38-Ø45 RDR NAME OF DESTINATION NODE
RDRDTUID DC    CL8' '              Ø46-Ø53 RDR NAME OF DESTINATION USER
RDRDPRGN DC    CL2Ø' '             Ø54-Ø73 RDR PROGRAMMER NAME.
RDRDROOM DC    CL8' '              Ø74-Ø81 RDR ROOM NUMBER.
RDRDDEPT DC    CL8' '              Ø82-Ø89 RDR DEPARTMENT NUMBER.
RDRDBLDG DC    CL8' '              Ø9Ø-Ø97 RDR BUILDING NUMBER.
         DC    CL2' '              Ø98-Ø99 FILLER
         DS    CL4                 1ØØ-1Ø3 FILLER.
         DC    CL4'    '           1Ø4-1Ø7 FILLER.
         DS    CL4                 1Ø8-111 FILLER.
         DS    CL8                 112-119 FILLER.
         DS    CL8                 12Ø-127 FILLER.
```

```
         DS    CL8                128-135 FILLER.
         DS    CL1                136-136 FILLER.
         DC    CL2' '             137-138 FILLER.
         DS    CL2                139-140 FILLER.
RDRGUS   DC    CL8' '             141-148 RDR USER ID.
         DS    CL2                149-150 FILLER.
*
PUNREC   DS    0CL150         001-150 PUN SETUP RECORD.
PUNRCDE  DS    C                  001-001 PUN RECORD CODE. (5).
PUNJOB   DC    CL8' '             002-009 PUN JOB NAME.
PUNGCL   DC    C' '               010-010 PUN OUTPUT CLASS.
PUNGPW   DC    CL8' '             011-018 PUN PASSWORD.
PUNDDP   DC    C' '               019-019 PUN OUTPUT DISPOSITION.
PUNDPR   DC    C' '               020-020 PUN OUTPUT PRIORITY.
PUNDSID  DC    C' '               021-021 PUN OUTPUT SYSTEM IDENTIFIER
PUNDUI   DC    CL16' '            022-037 PUN USER INFORMATION.
PUNDTNN  DC    CL8' '             038-045 PUN NAME OF DESTINATION NODE
PUNDTUID DC    CL8' '             046-053 PUN NAME OF DESTINATION USER
PUNDPRGN DC    CL20' '            054-073 PUN PROGRAMMER NAME.
PUNDROOM DC    CL8' '             074-081 PUN ROOM NUMBER.
PUNDDEPT DC    CL8' '             082-089 PUN DEPARTMENT NUMBER.
PUNDBLDG DC    CL8' '             090-097 PUN BUILDING NUMBER.
PUNONCPY DC    CL2' '             098-099 PUN NUMBER OF COPIES.
         DS    CL4                100-103 PUN NAME OF COMPACTION TABLE
PUNOFORM DC    CL4'    '          104-107 PUN FORM NUMBER. (PART 1).
         DS    CL4                108-111 PUN FORM NUMBER. (PART 2).
         DS    CL8                112-119 PUN EXTERNAL WRITER SUB SYST
         DS    CL8                120-127 FILLER.
         DS    CL8                128-135 FILLER.
         DS    CL1                136-136 FILLER.
PUNONSEP DC    CL2' '             137-138 PUN NUMBER OF SEPARATOR CARD
         DS    CL2                139-140 FILLER.
PUNGUS   DC    CL8' '             141-148 PUN USER ID.
         DS    CL2                149-150 FILLER.
*
LSTREC   DS    0CL150         001-150 LST SETUP RECORD.
LSTRCDE  DS    C                  001-001 LST RECORD CODE. (7).
LSTJOB   DC    CL8' '             002-009 LST JOB NAME.
LSTGCL   DC    C' '               010-010 LST OUTPUT CLASS.
LSTGPW   DC    CL8' '             011-018 LST PASSWORD.
LSTDDP   DC    C' '               019-019 LST OUTPUT DISPOSITION.
LSTDPR   DC    C' '               020-020 LST OUTPUT PRIORITY.
LSTDSID  DC    C' '               021-021 LST OUTPUT SYSTEM IDENTIFIER
LSTDUI   DC    CL16' '            022-037 LST USER INFORMATION.
LSTDTNN  DC    CL8' '             038-045 LST NAME OF DESTINATION NODE
LSTDTUID DC    CL8' '             046-053 LST NAME OF DESTINATION USER
LSTDPRGN DC    CL20' '            054-073 LST PROGRAMMER NAME.
LSTDROOM DC    CL8' '             074-081 LST ROOM NUMBER.
LSTDDEPT DC    CL8' '             082-089 LST DEPARTMENT NUMBER.
```

```
LSTDBLDG DC    CL8' '              Ø9Ø-Ø97 LST BUILDING NUMBER.
LSTONCPY DC    CL2' '              Ø98-Ø99 LST NUMBER OF COPIES.
         DS    CL4                 1ØØ-1Ø3 LST NAME OF COMPACTION TABLE
LSTOFORM DC    CL4'    '           1Ø4-1Ø7 LST FORM NUMBER. (PART 1).
         DS    CL4                 1Ø8-111 LST FORM NUMBER. (PART 2).
LSTOEWTR DS    CL8                 112-119 LST EXTERNAL WRITER SUB-SYST
LSTOFCB  DC    CL8' '              12Ø-127 LST FCB IMAGE PHASE.
LSTOUCB  DC    CL8' '              128-135 LST UCB IMAGE PHASE.
LSTOUCBO DC    C' '                136-136 LST UCB OPTIONS.
LSTONSEP DC    CL2' '              137-138 LST NUMBER OF SEPARATOR PAGE
         DS    CL2                 139-14Ø FILLER.
LSTGUS   DC    CL8' '              141-148 LST USER ID.
LSTCCODE DC    C' '                149-149 LST COMMAND CODE. (ASA/MACHI
LSTCVASA DC    C' '                15Ø-15Ø LST CONVERT ASA TO MACHINE.
*
         DS    ØD                  MUST BE DOUBLE WORD ALIGNED.
POWCPU   DS    ØCL1Ø
         DS    PL8
POWPID   DC    X'FFFF'
*
HØ8Ø     DC    H'8Ø'
H128     DC    H'128'
H132     DC    H'132'
ADDRESS  DC    F'Ø'
ASENDBUF DC    A(SENDBUF)
SVØØ15   DC    18F'Ø'
SVØ3     DC    F'Ø'
SV1ØA    DC    F'Ø'
SV1ØB    DC    F'Ø'
SV11A    DC    F'Ø'
SV11B    DC    F'Ø'
SV11C    DC    F'Ø'
SV11D    DC    F'Ø'
SV13     DC    F'Ø'
SV15     DC    F'Ø'
SAVEAREA DC    18F'Ø'
ADDRCSA  DC    F'Ø'
*
WAITLIST DC    A(INTECB)           INTECB = 1ST ELEMENT OF WAITLIST.
LISTCECB DC    A(Ø)                IJBXCECB = 2ND ELEMENT OF
                                   WAITLIST.
LISTEND  DC    X'FFFFFFFF'         INDICATE END OF WAITLIST
INTECB   DC    F'Ø'                ECB USED TO WAIT FOR TIMER
                                   INTERVALS
*
         DS    ØD
LSTCCB   CCB   SYSLST,LSTCCW,X'Ø4ØØ'
         DS    ØD
```

```
LSTCCW   CCW   X'Ø9',LSTBUF,X'2Ø',L'LSTBUF
         DC    C' '                DON'T MOVE/REMOVE THIS STATEMENT.
LSTBUF   DC    CL133' '            SYSLST BUFFER
*
OWNSPL   PWRSPL TYPE=GEN,PRFX=OWN,
               USERID=DPPOWP RØØØ *,ØØØ (*,ØØØ) ANY ØØØ * LOCAL
*
DPPOWPM  DC    X'FE'
*
         LTORG
*
XPCCB    XPCCB APPL=DPPOWP,TOAPPL=SYSPWR,
               BUFFER=(SENDBUF,2368),REPAREA=(REPLBUF,5ØØ)
REPLBUF  DS    CL5ØØ               BUFFER FOR RECEIVING DATA FROM
                                   VSE/P
         DC    C'SENDBUF'
         DS    ØF
SENDBUF  DS    ØCL2368             BUFFER FOR SENDING DATA TO
                                   VSE/POWER
         DS    CL2Ø96
SENDEND  DS    CL136
         DC    136X'ØØ'
*
DPPOWPE  DC    X'FF'
*
OWNSPLDS PWRSPL TYPE=MAP
*
         MAPXPCCB ,
*
* SYSTEM COMMUNICATIONS REGION. (SYSCOM).
*
SYSCOM   SYSCOM ,
*
* PARTITION COMMUNICATIONS REGION. (MAPCOMR).
*
         MAPCOMR ,
*
         END
```

Robert Botsis
Senior Systems Programmer (USA)

REXX/VSE to LE/VSE interface

We installed VSE/ESA Version 2 Release 1 last year, and are now using REXX/VSE (part of the central functions of VSE/ESA) and Language Environment for VSE/ESA (LE/VSE).

LE/VSE provides some very useful callable services for REXX/VSE users. Both COBOL for VSE/ESA and PL/I for VSE/ESA are LE/VSE-conforming languages, and can use these LE/VSE services directly. I have written the following interface routine for REXX/VSE, which allows REXX/VSE programs to use any LE/VSE callable service.

```
         TITLE 'REXXLE - REXX/VSE-LE/VSE-INTERFACE'
         EJECT
*********************************************************************
*
* The REXXLE program is an interface between REXX/VSE and LE/VSE.
* It allows LE/VSE callable services to be used within a
* REXX/VSE procedure, and is called by the following command:
*
*   address LINKPGM "REXXLE lepgm p1 p2 ... pn"
*
* The parameter lepgm must be a length-prefixed character string
* representing the name of the callable service.
*
* The other parameters are the parameters of the called service.
* The REXX/VSE procedure must initialize them before calling the
* service. See the LE/VSE Programming Guide for a description of the
* parameters. Output parameters must have the correct length.
*
* The following example uses the CEEDAYS service. The string
* Ø6/Ø2/1988, representing Ø2 June 1988, is converted into the Lilian
* format.
* The result of the last command (say lilian) is 148155.
*
*  prog = d2c(7,2) || "CEEDAYS"
*  datstring = d2c(1Ø,2) || "Ø6/Ø2/1988"
*  datpic    = d2c(1Ø,2) || "MM/DD/YYYY"
*  lilian = "    "
*  fc = "            "
*  address LINKPGM "REXXLE prog datstring datpic lilian fc"
```

```
*  lilian = c2d(lilian)
*  say lilian
*
*********************************************************************
         EJECT
*********************************************************************
* REGISTERS
*    15  ENTRY POINT ADDRESS
*    14  RETURN ADDRESS
*    13  ADDRESS OF THE SAVE AREA / DYNAMIC STORAGE AREA (DSA)
*    12  COMMON ANCHOR AREA (CAA) ADDRESS
*    11  BASE REGISTER
*    1Ø
*    9
*    8
*    7
*    6
*    5
*    4
*    3   ADDRESS OF THE PARAMETER LIST
*    2   CALLABLE LE/VSE-SERVICE, ADDRESS OF THE PARAMETER LIST
*    1   ADDRESS OF THE PARAMETER LIST
*    Ø
*********************************************************************
         EJECT
*********************************************************************
*        LE/VSE PROLOG
*********************************************************************
REXXLE  CEEENTRY MAIN=YES,                                          *
               AUTO=WORKSIZE
         USING WORKAREA,13             DYNAMIC STORAGE AREA
         LA    2,4(,1)                 PARAMETER LIST LE/VSE SERVICE
         L     3,Ø(,1)                 NAME OF THE SERVICE
         EJECT
*********************************************************************
*        LOAD AND CALL THE SERVICE
*********************************************************************
         CEELOAD NAMEADDR=(3),                                      *
               MF=(E,LOADDYN)
         LR    1,2                     LE/VSE SERVICE, PARAMETER LIST
         BALR  14,15                   CALL LE/VSE SERVICE
         EJECT
*********************************************************************
*        TERMINATE LE/VSE
*********************************************************************
         CEETERM RC=Ø
         EJECT
```

```
*********************************************************************
*        PROGRAM PROLOG AREA
*********************************************************************
PPA      CEEPPA
         EJECT
*********************************************************************
*        WORKAREA INCLUDING DYNAMIC SAVE AREA
*********************************************************************
WORKAREA DSECT
         ORG   *+CEEDSASZ            SPACE FOR THE DYNAMIC SAVE AREA
LOADDYN  CEELOAD NAMEADDR=(3),                                      *
               MF=L
         DS    ØD
WORKSIZE EQU   *-WORKAREA
         EJECT
*********************************************************************
*        DYNAMIC SAVE AREA, PART OF DSECT WORKAREA
*********************************************************************
         CEEDSA ,                   MAPPING OF THE DYNAMIC SAVE AREA
         EJECT
*********************************************************************
*        COMMON ANCHOR AREA
*********************************************************************
         CEECAA ,                     MAPPING OF COMMON ANCHOR AREA
         END REXXLE
```

For a description of the macros used in this program, see the *LE/VSE Programming Guide*. This also lists the callable services and their parameters.

When I tested the program, I got a problem with the CEELOAD macro. When using the execute form of the macro (MF=(E,addr)) to generate re-entrant code, other operands like FEEDBACK= were ignored. Although this was a known problem for VM/ESA and MVS/ESA, it was new for VSE/ESA (see APAR PQ00422). When the corresponding PTF is available, the program should check the feedback token after loading the callable service.

Walter Richters
(Germany)

SENDFVSE – transfer utility using reader

Although many companies today use both VM and VSE with relatively few problems, their files cannot be easily shared. The program presented here helps to break down the barriers, transferring files via a READER, with almost no limitations on record length.

SENDFVSE

```
SENDFVSE EXEC
/*
 Sends files from CMS to VSE environment.
 General information:
 1. User should type file-name, file-type, and file-mode of CMS file.
 2. The file to be sent is read from file-mode; each record is split
    into 'n' records of 8Ø bytes.
 3. A job is created with split records canned.
 4. In the VSE environment, the record is rebuilt and written as a
    SAM/VSAM file. It can then be read by another program or by
    utilities like VSAM/REPRO or SORT.
*/
parse   upper arg    fil_fn fil_ft fil_fm '(' option
parse   upper source . . fn     ft     fm syn .
signal  on    syntax
address command

       if fil_fn  = '?'  fil_fn = ''                then HELP()

  else if fil_fn ¬= ''  & fil_ft ¬= '' & fil_fm ¬= '' then nop

       else HELP()

INIT()

CHECK_FILE()

WRITE_JOB()

SUBMIT()

GOBACK: 'PULL SET'               /* Restore settings...          */

exit                             /* Return to CMS                */
```

```
/*-------------------------------------------------------------------*/
INIT:

'PUSH SET'                          /* Save CP SET status.           */
'CP   SET EMSG OFF'
'CP   SET IMSG OFF'

eof       = 'END-OF-FILE'
target    = 'VSETESTE'              /* VSE machine name, destination */
targetcat = 'TESTE'
dummy     = ' '                     /* User may ask for DUMMY file.  */
s         = '       '               /* Seven blanks...               */

return ''

/*-------------------------------------------------------------------*/
CHECK_FILE:

if option = 'DUMMY'
   then do
        len_rec = 8Ø
        recs    = '1'
        dummy   = 'X'
        return ''
       end

'LISTFILE' fil_fn fil_ft fil_fm '(STACK DATE' /* Locate file...    */

      if rc=Ø  then ATTRIBUTES()
 else if rc=28
         then do
              msg = 'File' fil_fn fil_ft fil_fm 'not found.'
              error()
             end
      else do
           msg = 'Reading failure:' fil_fn fil_ft fil_fm '- RC='rc'.'
           error()
          end

return ''

/*-------------------------------------------------------------------*/
ATTRIBUTES:

pull . . . tipo_arq len_rec recs . datev timev

datev = right(datev,8)
```

```
   mm = substr(datev,1,2)
   mm = translate(mm,Ø,' ')
   dd = substr(datev,4,2)
   yy = substr(datev,7,2)
datev = yy'/'mm'/'dd

return ''

/*-------------------------------------------------------------------*/
WRITE_JOB:

nrecs = right(recs,8,Ø)
  if dummy = '' then info = 'OF:' datev timev
else  info = 'DUMMY'

dlbl =     "// DLBL    FILOUT,'"
dlbl = dlblfil_fn'.'fil_ft
dlbl = dlbl"',,VSAM,DISP=(,KEEP),CAT="targetcat','
dlbl = left(dlbl,71)'*'

W("* $$ JOB JNM=SENDFVSE,LDEST=(,"userid()"),CLASS=Ø")
W("//     JOB SENDFVSE" fil_fn fil_ft info 'recsize:' len_rec)
W("// ON   $ABEND  GOTO ABEND ")
W("// ON   $CANCEL GOTO ABEND ")
W("// OPTION LINK             ")
W("// ASSGN  SYSLST,IGN       ")  /* Ignore COBOL listing....      */
W("// ASSGN  SYSØ2Ø,READER    ")
W(dlbl)
W("               RECORDS="recs",RECSIZE="len_rec) /* 15 blanks    */
W("// EXEC    FCOBOL          ")

WRITECOBOL()                       /* COBOL is dynamically created  */

W("/*                          ")
W("// EXEC LNKEDT              ")
W("// EXEC                     ")
W("SENDVSEHEADER"dummy nrecs left(userid(),8) date(o) time() datev
 timev)

if dummy = '' then WRITE_FILE()   /* Include target-file here      */

W("/*                          ")
W("// ASSGN SYSLST,ØØE         ")
W("// GOTO  END                ")
W("/. ABEND                    ")
W("/* PUT AN ACTION HERE...    ")  /* In case of ABEND, what to do ?*/
W("// GOTO  $EOJ               ")
W("/. END                      ")
```

```
W("/&                          ")
W("* $$ EOJ                    ")

return ''

/*-----------------------------------------------------------------*/
WRITE_FILE:

 do i=1 to recs
   'EXECIO 1 DISKR' fil_fn fil_ft fil_fm '(FINIS VAR LX'
    SPLIT_REC()
end

W(eof)

return ''

/*-----------------------------------------------------------------*/
SPLIT_REC:

left = 1
recl = len_rec

 do until recl <= Ø
    reg = substr(LX,left,8Ø)
    W(reg)
    left = left + 8Ø
    recl = recl - 8Ø
end

return ''

/*-----------------------------------------------------------------*/
WRITECOBOL:
                                /*---Part Ø1---------------------*/

 W(" CBL APOST                                                  ")
 W(s"IDENTIFICATION    DIVISION.                                ")
 W(s"PROGRAM-ID.       SENDFVSE.                                ")
 W(s"ENVIRONMENT       DIVISION.                                ")
 W(s"CONFIGURATION     SECTION.                                 ")
 W(s"INPUT-OUTPUT      SECTION.                                 ")
 W(s"FILE-CONTROL.                                              ")
 W(s"    SELECT        FILIN    ASSIGN TO SYSØ2Ø-UR-254ØR-S.    ")
 W(s"    SELECT        FILOUT   ASSIGN TO SYSØ21-DA-335Ø-S-FILOUT. ")
 W(s"I-O-CONTROL.                                               ")
 W(s"DATA              DIVISION.                                ")
 W(s"FILE              SECTION.                                 ")
```

```
 W(s"FD  FILIN                                                    ")
 W(s"    LABEL          RECORD   OMITTED                         ")
 W(s"    RECORD         CONTAINS 8Ø CHARACTERS.                  ")
 W(s"Ø1  REC-FILIN      PIC X(8Ø).                               ")
 W(s"FD  FILOUT                                                  ")
 W(s"    RECORD         CONTAINS" len_rec "CHARACTERS            ")
 W(s"    RECORDING      MODE     IS  F                           ")
 W(s"    LABEL RECORDS ARE       STANDARD                        ")
 W(s"    DATA  RECORD  IS        REC-FILOUT.                     ")
 W(s"Ø1  REC-FILOUT     PIC X("len_rec").                        ")
 W(s"WORKING-STORAGE    SECTION.                                 ")
 W(s"Ø1  WORKIN.                                                 ")
 W(s"    Ø2 EOF         PIC X(13).                               ")
 W(s"    Ø2 FILLER      PIC X(67).                               ")
 W(s"Ø1  REC-HEADER.                                             ")
 W(s"    Ø2 PATTERN     PIC X(13).                               ")
 W(s"    Ø2 DUMMY       PIC X.                                   ")
 W(s"    Ø2 FILLER      PIC X.                                   ")
 W(s"    Ø2 NUM-RECS    PIC 9(8).                                ")
 W(s"    Ø2 FILLER      PIC X.                                   ")
 W(s"    Ø2 USER        PIC X(8).                                ")
 W(s"    Ø2 FILLER      PIC X.                                   ")
 W(s"    Ø2 WHEN-SNT    PIC X(8).                                ")
 W(s"    Ø2 FILLER      PIC X.                                   ")
 W(s"    Ø2 TIME-SNT    PIC X(8).                                ")
 W(s"    Ø2 FILLER      PIC X.                                   ")
 W(s"    Ø2 WHEN-CMS    PIC X(8).                                ")
 W(s"    Ø2 FILLER      PIC X.                                   ")
 W(s"    Ø2 TIME-CMS    PIC X(8).                                ")
 W(s"Ø1  WORK.                                                   ")

left = 1
recl = len_rec
r=Ø

  if dummy = ''
     then do until recl <= Ø
             r=r+1
             if recl < 8Ø then reg = "   Ø2 REG"r"   PIC X("recl")."
             else              reg = "   Ø2 REG"r"   PIC X(8Ø)."
          W(s reg)
            left = left + 8Ø
            recl = recl - 8Ø
        end
else W(s"    Ø2  REGX  PIC X(8Ø).                                ")

                                  /*---Part Ø2---------------------*/
```

```
 W(s"Ø1  ALL-READ      PIC X       VALUE 'Ø'.                      ")
 W(s"Ø1  CONT-RECS     PIC 9(8)    VALUE  Ø.                       ")
 W(s"Ø1  RC            PIC S9(ØØ4) VALUE +Ø.                       ")
 W(s"PROCEDURE DIVISION.                                           ")
 W(s"BEGIN.   OPEN     INPUT   FILIN                               ")
 W(s"         OPEN     OUTPUT  FILOUT.                             ")
 W(s"         PERFORM  PROCESS-HEADER                              ")
 W(s"         PERFORM  REDO-RECORD    THRU REDO-RECORD-END        ")
 W(s"                  UNTIL ALL-READ EQUAL '1'.                   ")
 W(s"         DISPLAY  '..... WRITTEN RECORDS: ' CONT-RECS        ")
 W(s"                  UPON  CONSOLE.                              ")
 W(s"      IF NUM-RECS NOT EQUAL CONT-RECS THEN PERFORM CANCEL-JOB.")
 W(s"EOBEGIN. CLOSE  FILIN FILOUT STOP RUN.                        ")
 W(s"REDO-RECORD.                                                  ")

                                 /*---Part Ø3---------------------*/

 do g=1 to r
    rec="       PERFORM READIN THRU READINEND MOVE WORKIN TO REG"g
    W(s rec)
end

 W(s"           IF ALL-READ = Ø THEN                               ")
 W(s"           PERFORM WRITE-FILOUT THRU WRITE-FILOUT-END.        ")
 W(s"REDO-RECORD-END. EXIT.                                        ")
 W(s"READIN.    IF ALL-READ = Ø        THEN                       ")
 W(s"                                  READ FILIN INTO WORKIN.    ")
 W(s"           IF EOF = 'END-OF-FILE' THEN                        ")
 W(s"                                  MOVE '1' TO ALL-READ.      ")
 W(s"READINEND. EXIT.                                              ")
 W(s"WRITE-FILOUT. WRITE REC-FILOUT FROM WORK.                     ")
 W(s"              ADD  1              TO CONT-RECS.               ")
 W(s"WRITE-FILOUT-END. EXIT.                                       ")
 W(s"CANCEL-JOB.                                                   ")
 W(s"       DISPLAY '..... INPUT FILE INCORRECT.'      UPON CONSOLE")
 W(s"       MOVE 16 TO RC  CALL 'RETCODE' USING RC.                ")
 W(s"PROCESS-HEADER.                                               ")
 W(s"       PERFORM READIN THRU   READINEND                        ")
 W(s"               MOVE   WORKIN TO REC-HEADER.                   ")
 W(s"   IF PATTERN  NOT EQUAL 'SENDVSEHEADER'                      ")
 W(s"      DISPLAY '..... INVALID HEADER'              UPON CONSOLE")
 W(s"      PERFORM THEN CANCEL-JOB.                                ")
 W(s"   DISPLAY '..... SENT RECORDS   : '  NUM-RECS    UPON CONSOLE")
 W(s"   DISPLAY '..... BY USER        : '  USER        UPON CONSOLE")
 W(s"   DISPLAY '..... OF: ' WHEN-SNT ' AT: ' TIME-SNT UPON CONSOLE")
 W(s"   IF DUMMY = 'X' THEN                                        ")
 W(s"      DISPLAY '..... DUMMY FILE '                 UPON CONSOLE")
 W(s"                    PERFORM EOBEGIN.                          ")
```

```
 W(s"   DISPLAY '..... CMS FILE       :'               UPON CONSOLE")
 W(s"   DISPLAY '..... OF: ' WHEN-CMS ' AT: ' TIME-CMS UPON CONSOLE")

return ''

/*-----------------------------------------------------------------*/
W: parse arg register

queue register

return ''

/*-----------------------------------------------------------------*/
SUBMIT:

'CP SPOOL PUNCH TO *' target
'EXECIO' queued()   ' PUNCH'
'CP CLOSE PUNCH     '
'CP SPOOL PUNCH OFF '

'VMFCLEAR'
say '* * * -----------------SENDFVSE Utility----------------- * * *'
say 'CMS/FILE:' fil_fn fil_ft fil_fm
say '         ' info
say '          RECSIZE:' len_rec '- WITH:' recs 'records'
say '          was sent to' target 'machine.'
say '* * * ------------------------------------------------- * * *'

return ''

/*-----------------------------------------------------------------*/
ERROR:
say msg
'CP SPOOL ØØD PURGE'
GOBACK()
return

/*-----------------------------------------------------------------*/
SYNTAX:  sigl = sigl - 1
say '==> Syntax error at line' sigl
'CP SPOOL ØØD PURGE'
GOBACK()
return

/*-----------------------------------------------------------------*/
HELP:

'VMFCLEAR'
```

```
say 'Function: transfer utility of files from CMS to VSE environment'
say '          via READER with no restriction on length.            '
say ' '
say '  Format: SENDFVSE fn ft fm  ( DUMMY                           '
say ' '
say '   Obs.: DUMMY may be specified when dummy file is necessary.  '
say '          File will be created with no records.                '
exit

return ''

RETCODE SUBROUTINE

* $$ JOB  RETCODE,,9,Ø
* $$ PUN  DISP=I,CLASS=Ø
// JOB    RETCODE
// OPTION DECK
// OPTION LIST,LOG,SYM,DECK
// EXEC   ASSEMBLY
          PUNCH '// JOB    CATAL RETCODE'
          PUNCH '// EXEC   LIBR'
          PUNCH ' ACC S=PRD2.CONFIG'
          PUNCH ' CATALOG RETCODE.OBJ  REPLACE=YES'
          TITLE ' * * * RETCODE - SUBROUTINE * * *'
RETCODE  CSECT
         SAVE  (14,12)
         LR    5,15
         USING RETCODE,5
         L     4,Ø(,1)
         PACK  RTCDP,Ø(4,4)
         CVB   8,RTCDP
         SR    15,15
         LR    15,8
         EOJ   RC=(15)
         DS    ØD
RTCDP    DS    PL8
         END   RETCODE
/*
// EXEC ASSEMBLY
        PUNCH '/+'
        PUNCH '/*'
        PUNCH '/&&'
        PUNCH '* $$ EOJ'
        END
/*
/&
* $$ EOJ
```

Ricardo Quintal Reiche
Senior Systems Programmer (Brazil)

PSF enhancement

IBM's PSF/VSE development team recently developed PTF UD50347, which contains, in addition to the latest cumulative service, an APAR that makes life easier for programmers, systems staff, and operators.

The APAR, DY44439, introduces a new SET command which controls the Reasonable Resource Loading Value (RRLV) for the pagedefs and formdefs that PSF/VSE maintains across job boundaries. This means that shops can choose to have PSF load resources fresh from the VSE library for each print job, instead of reusing resources of the same name that may be lingering in storage from a previous print job. The net result is that, as resources are modified – for example in a development testing environment – new resources can be picked up immediately instead of having to restart the printer or rename the resource to a name that PSF has not yet processed. This is a major improvement for PSF/VSE application developers.

The SET command is an operator partition command entered using the MSG command with the DATA= parameter. In the example below, resources are set to zero; this means that PSF will delete the resources from PSF virtual storage once the last page of the print job has been sent to the printer. In effect, this causes PSF to reload the resources from the VSE library if and when they are needed in a subsequent print job. DEVNAME is the name of the printer as stated on the PSF PRINTDEV statement.

```
MSG H1,DATA=SET devname FORMDEF=Ø, PAGEDEF=Ø
```

The RRLV value can be any number between 0 and 32767. Note that the continual resource reloading caused by a low or zero setting may slightly decrease printer performance; this may not be suitable for a production environment. Similarly, a high setting will use more virtual storage in the PSF partition.

Our site uses PSF Version 2.2.1 under VSE/ESA 2.1.0, and the APAR works fine with our 3160 printers. However, it doesn't work with 3916 printers, which is what our developers use for testing – the PSF support group is working on a solution for this. Other printers may

also fall into the not-supported category, but a quick check with IBM will give you the latest details.

Steve Bernard
(USA)

Performance hints and tips

Xephon is hoping to publish a number of hints and tips on how to improve the performance of VSE and the subsystems running under it. Send us your tips on:

- Performance management
- Improving hardware performance
- Enhancing software performance
- Reducing paging and swapping
- Improving I/O
- Monitoring and controlling network performance
- Subsytem performance
- Tuning
- Capacity planning.

Articles on these (and any other) subjects can be sent to the editor, Fiona Hewitt, at any of the addresses shown on page 2, or sent using the Internet to 100336.1412@compuserve.com.

Remember that we pay $250 (£170) per 1,000 words and $140 (£90) per 100 lines of code published if you give us copyright. If you prefer to retain copyright and grant us a licence to publish instead, we pay $125 (£85) per 1,000 words and $70 (£45) per 100 lines of code.

We welcome both short 'hints and tips' articles and longer articles with example code.

An easy VSE to MVS connection

This article describes the conversion of our VSE system to MVS.

The first step was to connect the two systems together using VSE/POWERNET and JES2/NJE. This meant that our conversion team could continue to use VM/XEDIT for file editing and JCL coding. Other benefits included direct JCL submission from VM to JES2, and the fact that SYSOUT could be routed back between the two systems.

Both POWERNET and JES2/NJE come with the base operating system. We connected the two systems using two communication controllers, although you could also use one controller with a two-channel adapter.

Figure 1 shows a sample communication network configuration.

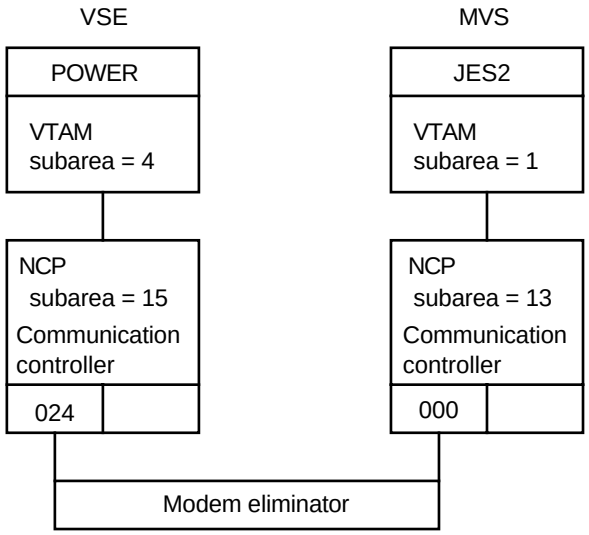

Figure 1: Sample communication network configuration

VM/VSE SYSTEM SET-UP

The VM/VSE system set-up is as follows:

1 Assemble the POWERNET module into the user library.

```
VSENDT     PNODE NODE=VSE,APPLID=VSEA,LOCAL=YES
           PNODE NODE=MVS,APPLID=MVSA
           END
```

2 Code the PNET name on your POWER macro, and assemble it into the user library. Place a LIBDEF statement for the user library before // EXEC VSEPOWER in your ASI. You need to IPL this new ASI to make it effective.

```
VSEPOWER    POWER DBLK=Ø
                   .
                  (your other VSE power macro goes here)
                   .
                  PNET=VSENDT
                   .
            END
```

3 Code and assemble the MODETAB table for POWERNET entry PNETNJE0.

```
MODETAB1 MODETAB
IBMPOWER MODEENT LOGMODE=IBMPOWER,FMPROF=X'Ø3',TSPROF=X'Ø3',
                 PRIPROT=X'72',SECPROT=X'72',COMPROT=X'2ØØØ',
                 PSNDPAC=X'Ø3',SRCVPAC=X'Ø3',SSNDPAC=X'Ø3'
PNETNJEØ MODEENT LOGMODE=PNETNJEØ,SRCVPAC=3,PSNDPAC=3,SSNDPAC=3
         MODEEND
```

4 Set up VTAM application minor node. The minor node name should be included in the ATCCONxx table as well as in the cross domain manager (CDRM) and resources (CDRSC).

```
VSEAPPL  VBUILD TYPE=APPL
VSECICS  APPL AUTH=(PASS,ACQ,NVPACE)
VSE      APPL AUTH=(BLOCK,PASS,ACQ),DLOGMOD=IBMPOWER,
              VPACING=3,MODETAB=MODETAB1
VSEA     APPL AUTH=(ACQ),DLOGMOD=PNETNJEØ,
              VPACING=3,MODETAB=MODETAB1

CDRM     VBUILD TYPE=CDRM
VSE      CDRM  SUBAREA=4,CDRDYN=YES,CDRSC=OPT
MVS      CDRM  SUBAREA=1,CDRDYN=YES,CDRSC=OPT

CDRSC    VBUILD TYPE=CDRSC
VSECICS  CDRSC CDRM=VSE
MVSCICS  CDRSC CDRM=MVS
TSO      CDRSC CDRM=MVS
MVSA     CDRSC CDRM=MVS
```

5 Set up PATH table for VTAM.

```
PATH4    PATH  DESTSA=(14,4,13,1)
               ERØ=(14,1)
               VRØ=Ø
```

6 Set up NCP parameters.

```
SSCP4    PCCU  AUTODMP=NO,
               .
               .
               SUBAREA=4
               .
PATH1    ADJSUB=13,DESTSUB=1
               .
SDLC   GROUP LNCTL=SDLC,
             DIAL=NO,
             REPLYTO=1.Ø,
             RETRIES=(7,3,5),
             TEXTTO=3.Ø,
             ISTATUS=ACTIVE,
             DUPLEX=FULL,
             VPACING=2,
             TYPE=NCP
NNØ3   LINE  ADDRESS=(Ø2A,Ø2B),
             POLLED=YES,           ¥
             CLOCKNG=EXT,
             PASSLIM=254,
             MAXOUT=7,
             NRZI=NO,
             NEWSYNC=NO,
             TRANSFR=32,
             SPEED=96ØØ,
             TADDR=ØD
V3PU   PU    ANS=CONT,
             PUTYPE=4,
             SUBAREA=13
```

MVS/JES2 SYSTEM SET-UP

The MVS/JES2 system set-up is as follows:

1 Code JES2 initialize parameters.

```
NJEDEF   DELAY=12Ø,
         JRNUM=1,
         JTNUM=1,
         LINENUM=4,
```

```
         NATNUM=150,
         NODENUM=2,
         OWNNODE=1,
         PATH=1,
         RESTMAX=8000000,
         RESTNODE=100,
         RESTTOL=0,
         SRNUM=1,
         STNUM=1
N1       NAME=MVS,SNA
N2       NAME=VSE,SNA
APPL     APPLID=MVSA,
         NODE=1
APPL     APPLID=VSEA,
         NODE=2
LOGON1   APPLID=MVSA
LINE1    UNIT=SNA
LINE2    UNIT=SNA
LINE3    UNIT=SNA
LINE4    UNIT=SNA
CONNECT  NODEA=1,MEMBA=1,
         NODEB=2,MEMBB=1,
         REST=150
```

2 Set up VTAM application minor node. The minor node name should be included in the ATCCONxx table as well as in the cross domain manager (CDRM) and resources (CDRSC).

```
MVS      VBUILD TYPE=APPL
MVS      APPL  AUTH=(ACQ,PASS,VPACE),VPACING=5

CDRM     VBUILD TYPE=CDRM
MVS      CDRM  SUBAREA=1,CDRDYN=YES,CDRSC=OPT,ISTATUS=ACTIVE
VSE      CDRM  SUBAREA=4,CDRDYN=YES,CDRSC=OPT,ISTATUS=INACTIVE

CDRSC    VBUILD TYPE=CDRSC
MVSCICS  CDRSC CDRM=MVS,ISTATUS=ACTIVE
VSECICS  CDRSC CDRM=VSE,ISTATUS=ACTIVE
VSEA     CDRSC CDRM=VSE,ISTATUS=ACTIVE
```

3 Set up VTAM path table.

```
PATH1    PATH  DESTSA=(15),              FOR VSE 3705.
               ER0=(13,1),VR0=0
         PATH  DESTSA=(13),              FOR MVS 3720
               ER0=(13,1),VR0=0
         PATH  DESTSA=(4),               FOR SUBAREA 4 - VSE
               ER0=(13,1),VR0=0
```

4 Set up NCP definitions.

```
SSCP1    PCCU  AUTODMP=NO,
                .
                .
                SUBAREA=1
                .
PATH1    PATH  DES*TSA=1,
               ERØ=(1,1),VRØ=Ø
PATH4    PATH  DESTSA=4,
               ERØ=(15,1),VRØ=Ø
PATH5    PATH  DESTSA=(15),
               ERØ=(15,1),VRØ=Ø
APRI     SDLCST GROUP=GRP1,MODE=PRI,
               RETRIES=(7,3,5),
               MAXOUT=7,
               PASSLIM=254
ASEC     SDLCST GROUP=GRP2,MODE=SEC,
               RETRIES=(7),
               MAXOUT=7,
               PASSLIM=254
GRP1     GROUP LNCTL=SDLC,MODE=PRI,ACTIVTO=12Ø
GRP2     GROUP LNCTL=SDLC,MODE=SEC
GRP3     GROUP LNCTL=SDLC,
               DIAL=NO,
               ACTIVTO=6Ø,
               REPLYTO=6Ø,
               TEXTTO=3.Ø,
               RETRIES=(7,3,5),
               ISTATUS=ACTIVE,
               DUPLEX=FULL,
               TYPE=NCP
NNØ4     LINE  ADDRESS=(Ø,FULL),
           DUPLEX=FULL,
           DATMODE=FULL,
           NRZI=NO,
           CLOCKNG=EXT,
           ISTATUS=ACTIVE,
           MAXOUT=7,
           PASSLIM=254,
           TRANSFR=32,                  MUST BE THE SAME AS VSE
           NEWSYNC=NO,
           SDLCST=(APRI,ASEC),
           SPEED=96ØØ
V4PU     PU    PUTYPE=4,
           ANS=CONT,
           TGN=1
```

OPERATIONAL INSTRUCTIONS

The operational instructions are as follows:

- To start POWERNET, use the following:

```
PSTART PNET,MVS
```

- To stop POWERNET, use the following:

```
PSTOP  PNET,MVS
```

- To start JES2/NJE, use the following:

```
$S LNE4
$S LOGON1
$SN,LNE4,A=VSEA
```

- To stop JES2/NJE, use the following:

```
$P LNE4
$P LOGON1
```

- To submit JCL from VSE to MVS, code XDEST=MVS on POWER JECL, and send the JCL to the VSE machine. POWER will transfer the JCL to its XMIT queue and transmit to MVS. When the JCL arrives at the MVS machine, JES2 will place it to the spool queue.

```
* $$ JOB JNM=SENDJES,DISP=D,CLASS=A,XDEST=MVS
   //IEBGENER JOB (1234),IPSAP00,USER=MYID,PASSWORD=MYPASS
                    .
                    .
   //
   /*
   /&
   * $$ EOJ
```

- To transfer SYSOUT from VSE to MVS, change the destination to MVS.
- To transfer SYSOUT from MVS to VSE, change the destination to VSE.

Dave Tang
(Canada)

VSE news

IBM has announced that its new Multiprise 2000 range of machines now supports VSE/ESA V1.3 and V1.4 running in 370 mode, which means that old applications can be preserved and migrated to the millennium-compliant Multiprises.

For further information, contact your local IBM representative.

* * *

IBM has announced OPTI-AUDIT for VSE Version 1.1.0, which enables you to perform a static analysis of VSE libraries and build an inventory of all programs contained on a VSE system. The program offering assists with migration from an earlier VSE/SP or VSE/ESA system to a current VSE/ESA Version 2 system. It monitors the execution of batch jobs and on-line transactions, extracting job/program/file cross-referencing information, and also provides a source scanning facility for COBOL source code. Finally, it produces a variety of reports to manage the Year 2000 conversion, including program status, program usage, file cross-referencing, and job cross-referencing. IBM OPTI-AUDIT for VSE Version 1.1.0 requires VSE/ESA V1.1 or later, VSE/ESA V2.1 or later, VSE/SP V3 or later, or VSE/SP V4 or later.

For more information, contact your local IBM representative.

* * *

IBM has announced general availability of its DB2 Server for VSE and VM Version 5 Release 1. New functions include a DUOW application server, DataPropagator-Relational capture support, an Uncommitted Read option, free DBAs, and VSE Extended User buffering.

For more information, contact your local IBM representative.

* * *

IBM has published the first edition of its *Native TCP/IP Solution for VSE* Redbook.

For more information, contact your local IBM representative.

* * *