28 VSE

December 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.

Synchronizing batch tasks

The two programs presented in this article were developed and tested under VSE/ESA Version 1.3. They are now running under VSE/ESA Version 2.2.

PROGRAM B238SYN

The first program, B238SYN, allows the synchronization of two or more batch partitions at job control level by simply coding, for example:

```
// EXEC B238SYN,PARM='VSE UPDATE'
```

in one partition and

```
// EXEC B238SYN,PARM='UPDATE VSE'
```

in another partition.

When one of the two partitions executes the above step, the task waits until the other partition executes the corresponding step. This is illustrated by the following example :

```
Clock      Partition A                    Partition B
-------    ----------------------------   ----------------------------
  ...      ...                            ...
  09:30    // EXEC B238SYN,PARM='A B'     ...
  ...      Wait state                     ...
  09:40    Removed from wait state        // EXEC B238SYN,PARM='B A'
  09:40    Next step                      Next step
  ...      ...                            ...
```

The PARM string of the EXEC statement usually consists of two items. Each item can be up to 8 bytes in length. Any character in the PARM string lower than A (X'C0') is treated as a delimiter. Leading and trailing delimiters are allowed. For example, the following two statements

```
// EXEC B238SYN,PARM='/XEPHON//NEWBURY/'
  // EXEC B238SYN,PARM='XEPHON NEWBURY'
```

will produce identical results.

The two items are used as the names of cross-partition communication

control blocks (XPCCBs). The first item is the control block name of the first side and must be unique in the VSE system. The second item is the control block name of the other side.

Invoking program B238SYN with only one item in the PARM string has a special meaning. For example, the statement

```
// EXEC B238SYN,PARM='COMMON'
```

will not set the partition itself into the wait state, but removes from the wait state all other partitions that are executing a step such as the following:

```
// EXEC B238SYN,PARM='MYXPCCB COMMON'
```

The following example shows how to release the job ENDDAY once the back-up jobs BACKUP1 and BACKUP2 have both finished.

```
// JOB BACKUP1                        // JOB BACKUP2
  -- Backup steps --                   -- Backup steps --
  // EXEC B238SYN,PARM='BUP1 BUP2'     // EXEC B238SYN,PARM='BUP2 BUP1'
  // PWR PRELEASE RDR,ENDDAY           /&
  /&
```

Because of restrictions on the EOJ macro with the RC keyword, the program can execute only below the 16 MB line (RMODE 24).

Program B238SYN simply checks the PARM string. If an error is detected, control is given back to job control with a return code 9; otherwise, control is given to subroutine B239SYN, which does all the synchronization work (see below).

B238SYN

```
TITLE 'B238SYN - SYNCHRONIZATION OF BATCH PARTITIONS'
B238SYN  CSECT
B238SYN  AMODE 31
B238SYN  RMODE 24
         EJECT
*********************************************************************
*        REGISTER EQUATES
*********************************************************************
RØ       EQU   Ø
R1       EQU   1
R2       EQU   2
R3       EQU   3
R4       EQU   4
R5       EQU   5
```

```
R6       EQU   6
R7       EQU   7
R8       EQU   8
R9       EQU   9
R1Ø      EQU   1Ø
R11      EQU   11
R12      EQU   12
R13      EQU   13
R14      EQU   14
R15      EQU   15
         EJECT
*********************************************************************
* REGISTER USAGE:
*
*    R15 PROGRAM ENTRY POINT, RETURN CODE
*    R14 RETURN ADDRESS
*    R13 SAVE AREA ADDRESS
*    R12
*    R11
*    R1Ø
*    R9  BASE REGISTER
*    R8
*    R7
*    R6
*    R5  WORK REGISTER
*    R4  WORK REGISTER
*    R3  WORK REGISTER
*    R2  FIRST CHARACTER PAST PARM STRING
*    R1  ADDRESS OF PARM STRING (INPUT PARAMETER)
*    RØ
*********************************************************************
         EJECT
*********************************************************************
*        TEST INPUT PARAMETER, INITIALIZE REGISTERS AND STORAGE
*********************************************************************
         BALR  R9,Ø                    LOAD BASE REGISTER
         USING *,R9                    ESTABLISH ADDRESSABILITY
         CR    R1,R15                  PARM STRING EXISTS
         BE    RETURN9                 NO, INFORM JOB CONTROL
         TM    Ø(R1),X'8Ø'             HIGH ORDER BIT OK
         BNO   RETURN9                 NO, INFORM JOB CONTROL
         LA    R13,SAVEAREA            ADDRESS OF SAVE AREA
         L     R1,Ø(,R1)               ADDRESS OF PARM STRING
         LH    R2,Ø(,R1)               LENGTH OF PARM STRING
         LA    R1,2(,R1)               FIRST CHARACTER OF PARM STRING
         LA    R2,Ø(R1,R2)             FIRST CHARACTER PAST PARM
                                        STRING
         XC    OTHXECB,OTHXECB         INITIALIZE SECOND NAME
         EJECT
```

```
*********************************************************************
*        EXTRACT ITEMS (CONTROL BLOCKS) FROM PARM STRING
*        FIRST SKIP LEADING DELIMITERS
*********************************************************************
STARTSEP DS    ØH
         CLI   Ø(R1),C'A'              DELIMITER
         BNL   TESTANG1                NO, FIRST NON-DELIMITER
         LA    R1,1(,R1)               NEXT CHARACTER
         CR    R1,R2                   IS THIS THE LAST CHARACTER
         BL    STARTSEP                NO, TEST NEXT CHARACTER
         B     RETURN9                 NO ITEM FOUND, INFORM JCL
         EJECT
*********************************************************************
*        EXTRACT FIRST ITEM FROM PARM STRING
*********************************************************************
TESTANG1 DS    ØH
         LR    R3,R1                   START FIRST ITEM
NOEND1   DS    ØH
         LA    R1,1(,R1)               NEXT CHARACTER
         CR    R1,R2                   IS THIS THE LAST CHARACTER
         BNL   MOVEERST                YES, MOVE ITEM
         CLI   Ø(R1),C'A'              ITEM DELIMITER
         BNL   NOEND1                  NO, PART OF FIRST ITEM
MOVEERST DS    ØH
         LR    R4,R1                   CHARACTER PAST FIRST ITEM
         SR    R4,R3                   LENGTH OF FIRST ITEM
         LA    R5,L'OWNXECB            GREATEST LENGTH
         CR    R4,R5                   FIRST ITEM TOO LONG
         BH    RETURN9                 YES, INFORM JOB CONTROL
         MVI   OWNXECB,C' '            FIRST NAME BLANK
         MVC   OWNXECB+1(L'OWNXECB-1),OWNXECB
         BCTR  R4,Ø                    LENGTH FOR EXECUTE
         EX    R4,MVCXECB1             MOVE FIRST ITEM TO FIRST NAME
         CR    R1,R2                   IS THIS THE LAST CHARACTER
         BNL   CALLSYN                 YES, INPUT OK
         EJECT
*********************************************************************
*        SKIP DELIMITER BETWEEN ITEMS
*********************************************************************
MITTLSEP DS    ØH
         LA    R1,1(,R1)               NEXT CHARACTER
         CR    R1,R2                   IS THIS THE LAST CHARACTER
         BNL   CALLSYN                 YES, INPUT OK
         CLI   Ø(R1),C'A'              ITEM DELIMITER
         BL    MITTLSEP                YES, TEST NEXT CHARACTER
         EJECT
*********************************************************************
*        EXTRACT SECOND ITEM FROM PARM STRING
*********************************************************************
         LR    R3,R1                   START SECOND ITEM
```

```
NOEND2   DS    ØH
         LA    R1,1(,R1)               NEXT CHARACTER
         CR    R1,R2                   IS IT THE LAST CHARACTER
         BNL   MOVEZWEI                YES, MOVE ITEM
         CLI   Ø(R1),C'A'              ITEM DELIMITER
         BNL   NOEND2                  NO, PART OF SECOND ITEM
MOVEZWEI DS    ØH
         LR    R4,R1                   CHARACTER PAST SECOND ITEM
         SR    R4,R3                   LENGTH OF SECOND ITEM
         LA    R5,L'OTHXECB            GREATEST LENGTH
         CR    R4,R5                   SECOND ITEM TOO LONG
         BH    RETURN9                 YES, INFORM JOB CONTROL
         MVI   OTHXECB,C' '            SECOND NAME BLANK
         MVC   OTHXECB+1(L'OTHXECB-1),OTHXECB
         BCTR  R4,Ø                    LENGTH FOR EXECUTE
         EX    R4,MVCXECB2             MOVE SECOND ITEM TO SECOND
                                        NAME
         CR    R1,R2                   IS IT THE LAST CHARACTER
         BNL   CALLSYN                 YES, INPUT OK
         EJECT
*********************************************************************
*        SKIP TRAILING DELIMITERS
*********************************************************************
ENDSEP   DS    ØH
         LA    R1,1(,R1)               NEXT CHARACTER
         CR    R1,R2                   IS IT THE LAST CHARACTER
         BNL   CALLSYN                 YES, INPUT OK
         CLI   Ø(R1),C'A'              DELIMITER
         BL    ENDSEP                  YES, TEST NEXT CHARACTER
         B     RETURN9                 MORE THAN TWO ITEMS, INFORM
                                        JCL
MVCXECB1 MVC   OWNXECB(Ø),Ø(R3)        MOVE FIRST ITEM (EXECUTE)
MVCXECB2 MVC   OTHXECB(Ø),Ø(R3)        MOVE SECOND ITEM (EXECUTE)
         EJECT
*********************************************************************
*        PERFORM SYNCHRONIZATION
*********************************************************************
CALLSYN  DS    ØH
         CALL  B239SYN,(INPPARM,OUTPARM)
         LH    R15,XPCCR15             RETURN CODE (JOB CONTROL)
         EJECT
*********************************************************************
*        TERMINATE PROGRAM WITH RETURN CODE
*********************************************************************
ENDOFJOB EOJ   RC=(15)
         EJECT
*********************************************************************
*        SET JCL RETURN CODE FOR MISSING OR WRONG PARAMETER
*********************************************************************
RETURN9  DS    ØH
```

```
         LA    R15,9                    SET JCL RETURN CODE 9
         B     ENDOFJOB                 TERMINATE PROGRAM
         EJECT
*************************************************************************
*        SAVE AREA
*************************************************************************
SAVEAREA DS    9D                       OWN SAVEAREA
         EJECT
*************************************************************************
*        PARAMETER AREA
*************************************************************************
INPPARM  DS    ØCL16                    FIRST PARAMETER B239SYN
OWNXECB  DS    CL8                      NAME OF OWN CONTROL BLOCK
OTHXECB  DS    CL8                      NAME OF PARTNER'S CONTROL
                                         BLOCK
         DS    ØH                       FORCE ALIGNMENT
OUTPARM  DS    ØCL4                     SECOND PARAMETER B239SYN
XPCCR15  DS    H                        XPCC RETURN CODE (REGISTER 15)
XPCCFKT  DS    XL1                      XPCC FUNCTION BYTE
XPCCRC   DS    XL1                      XPCC RETURN CODE (IJBXRETC)
         END   B238SYN
```

SUBROUTINE B239SYN

Two parameters must be passed.

The first parameter is an input parameter, and consists of two fields:

- The first field is an 8-byte character string, and must contain the name of the cross-partition communication control block (XPCCB). This name must be unique in the VSE system. Program B238SYN pads the first item of the PARM string to the right with blanks, and stores the result in this field.
- The second field is an 8-byte character string, and contains the name of the application to which the connection is to be established – ie the name of the partner's XPCCB. If the first byte of this field contains X'00', the task will not set itself into the wait state, but removes from the wait state all other tasks that called this subroutine with a second input field identical to the first input field of this actual call. When B238SYN is executed and there is a second item in the PARM string, this item is padded to the right with blanks and stored in this second field; otherwise, the field is initialized to binary 0.

The second parameter is an output parameter, and consists of three

fields. If a cross-partition communication control (XPCC) request fails, information about the first failing request is stored as described below.

- The first field is a halfword. It contains the return code of the failing XPCC request stored in register 15. Possible values, which are described in the *VSE/ESA System Macros User's Guide*, are as follows:
  - 0 (X'0000') – Success.
  - 4 (X'0004') – Same as 0. Additional return information is stored in the third field.
  - 8 (X'0008') – Request was rejected. The third field defines the reason for the failure.
  - 12 (X'000C') – Request was rejected. The XPCCB address or the length of XPCCB is invalid. This value should not occur.
- The second field is one byte in length, and contains the function byte of the failing XPCC request. Possible values are:
  - X'01' – IDENTIFY
  - X'02' – CONNECT
  - X'09' – DISCONNECT
  - X'0C' – TERMINATE
- The third field is one byte in length, and contains the return code of the failing XPCC request as stored in the IJBXRETC field of the cross-partition communication control block (XPCCB). Detailed information is supplied by mapping the MAPXPCCB macro.

If no problems occur, all the fields of the second parameter are initialized with binary 0.

Program B238SYN uses only the first field, which is passed to job control as return code.

The following example shows how to call B239SYN from a COBOL program:

```
...
  WORKING-STORAGE SECTION.
    ...
  Ø1  XECBAB.
      Ø2  MYXECB     PIC X(Ø8) VALUE 'JOBA'.
      Ø2  YOURXECB   PIC X(Ø8) VALUE 'JOBB'.
  Ø1  RETCOD1.
      Ø2  EOJRC1     PIC S9(4) COMP.
      Ø2  FUNKT1     PIC X(1).
      Ø2  XPCCR1     PIC X(1).
    ...
  PROCEDURE DIVISION.
    ...
    CALL 'B239SYN' USING XECBAB RETCOD1.
    ...
```

This call causes the task to wait until the next corresponding call of a second task:

```
...
  WORKING-STORAGE SECTION.
    ...
  Ø1  XECBBA.
      Ø2  MEINXECB   PIC X(Ø8) VALUE 'JOBB'.
      Ø2  DEINXECB   PIC X(Ø8) VALUE 'JOBA'.
  Ø1  RETCOD2.
      Ø2  EOJRC2     PIC S9(4) COMP.
      Ø2  FUNKT2     PIC X(1).
      Ø2  XPCCR2     PIC X(1).
    ...
  PROCEDURE DIVISION.
    ...
    CALL 'B239SYN' USING XECBBA RETCOD2.
    ...
```

The first task can also be removed from the wait state by one of the following job control statements:

```
// EXEC B238SYN,PARM='JOBB JOBA'
  // EXEC B238SYN,PARM='JOBB'
```

We call subroutine B239SYN directly as part of an interface program that loads members of our CA-VOLLIE editor into the GETVIS partition. Once a week, we run back-up and restore to reorganize the editor's file. If the required member is not on disk, this interface may not work during restore. In this case, the interface program calls B239SYN with the first input field piVOLLIE (string pi replaced by partition ID) and second input field VOLLIEND, and waits. This may happen in more than one batch partition.

The step which restores the editor's file is followed by

```
// EXEC B238SYN,PARM='VOLLIEND'
```

which removes all the above partitions (if any) from the wait state.

B239SYN

```
TITLE 'B239SYN - SYNCHRONIZATION OF BATCH TASKS'
B239SYN  CSECT
B239SYN  AMODE ANY
B239SYN  RMODE ANY
         EJECT
*************************************************************************
*                         PARAMETER-LIST
*************************************************************************
PARAM    DSECT
ADRXECB  DS    A                      ADDRESS FIRST PARAMETER
ADRRETC  DS    A                      ADDRESS SECOND PARAMETER
         EJECT
*************************************************************************
*        FIRST PARAMETER (INPUT PARAMETER, NAMES OF CONTROL BLOCKS)
*************************************************************************
         USING XECBNAME,R7
XECBNAME DSECT
OWNXECB  DS    CL8                    NAME OF OWN CONTROL BLOCK
OTHXECB  DS    CL8                    NAME OF PARTNER'S CONTROL
                                       BLOCK
         EJECT
*************************************************************************
*        SECOND PARAMETER (OUTPUT PARAMETER, RETURN CODES)
*************************************************************************
         USING RETCOD,R8
RETCOD   DSECT
XPCCR15  DS    H                      XPCC RETURN CODE (REGISTER 15)
XPCCFKT  DS    XL1                    XPCC FUNCTION BYTE
XPCCRC   DS    XL1                    XPCC RETURN CODE (IJBXRETC)
RETCODLG EQU   *-RETCOD               LENGTH OF OUTPUT PARAMETER
         EJECT
*************************************************************************
*        CROSS-PARTITION COMMUNICATION CONTROL BLOCK (DSECT)
*************************************************************************
         MAPXPCCB
         EJECT
*************************************************************************
*        REGISTER EQUATES
*************************************************************************
RØ       EQU   Ø
R1       EQU   1
```

```
R2       EQU   2
R3       EQU   3
R4       EQU   4
R5       EQU   5
R6       EQU   6
R7       EQU   7
R8       EQU   8
R9       EQU   9
R1Ø      EQU   1Ø
R11      EQU   11
R12      EQU   12
R13      EQU   13
R14      EQU   14
R15      EQU   15
         EJECT
B239SYN  CSECT
*********************************************************************
* REGISTER USAGE:
*
*    R15 PROGRAM ENTRY POINT, XPCC RETURN CODE
*    R14 RETURN ADDRESS
*    R13 SAVE AREA ADDRESS
*    R12
*    R11
*    R1Ø
*    R9  BASE REGISTER
*    R8  ADDRESS OF SECOND PARAMETER
*    R7  ADDRESS OF FIRST PARAMETER
*    R6  RETURN ADDRESS FOR INTERNAL SUBROUTINES (XPCC FUNCTION)
*    R5  XPCC FUNCTION BYTE
*    R4  CROSS-PARTITION COMMUNICATION CONTROL BLOCK
*    R3
*    R2
*    R1  ADDRESS OF PARAMETER-LIST, USED BY IBM MACROS
*    RØ  USED BY IBM MACROS
*********************************************************************
         EJECT
*********************************************************************
*        SET UP LINKAGE REGISTERS
*********************************************************************
         SAVE  (14,12)                SAVE CALLER'S REGISTERS
         BALR  R9,Ø                   LOAD BASE REGISTER
         USING *,R9                   ESTABLISH ADDRESSABILITY
         ST    R13,SAVEAREA+4         CALLING PROGRAM'S SAVE AREA
         LA    R13,SAVEAREA           ADDRESS OWN SAVE AREA
         EJECT
*********************************************************************
*        LOAD ADDRESS OF PARAMETERS
*********************************************************************
         USING PARAM,R1               ADDRESS PARAMETER-LIST
```

```
         L     R7,ADRXECB              ADDRESS OF FIRST PARAMETER
         L     R8,ADRRETC              ADDRESS OF SECOND PARAMETER
         DROP  R1
         EJECT
*************************************************************************
*        INITIALIZE CROSS-PARTITION COMMUNICATION CONTROL BLOCK
*************************************************************************
         XC    RETCOD(RETCODLG),RETCOD INITIALIZE OUTPUT PARAMETER
         LA    R4,BATXPCCB             ADDRESS OF CROSS-PARTITION
         USING IJBXPCCB,R4               COMMUNICATION CONTROL BLOCK
         MVC   BATXPCCB(IJBXEND-IJBXSTRT),KSTXPCCB
         MVC   IJBXAPPL,OWNXECB        OWN CONTROL BLOCK
         MVC   IJBXTOAP,OTHXECB        CONTROL BLOCK OF PARTNER
         CLI   IJBXTOAP,X'ØØ'          PARTNER ANY
         BNE   NOANY                   NO, PARTNER WITH WAIT
         XC    IJBXTOAP,IJBXTOAP       PARTNER ANY, NO WAIT
NOANY    DS    ØH
         EJECT
*************************************************************************
*        XPCC FUNCTION IDENTIFY (LOG ON)
*************************************************************************
         LA    R5,IJBXID               FUNCTION IDENTIFY
         MVI   IJBXFDSC,IJBXUNIQ       FUNCTION DESCRIPTOR UNIQUE
         BAL   R6,XPCCSERV             PERFORM XPCC FUNCTION
         EJECT
*************************************************************************
*        XPCC FUNCTION CONNECT (ESTABLISH COMMUNICATION PATH)
*************************************************************************
         LA    R5,IJBXCON              FUNCTION CONNECT
         BAL   R6,XPCCSERV             PERFORM XPCC FUNCTION
         EJECT
*************************************************************************
*        XPCC FUNCTION DISCONNECT (DISCONNECT COMMUNICATION PATH)
*************************************************************************
         LA    R5,IJBXDSC              FUNCTION DISCONNECT
         BAL   R6,XPCCSERV             PERFORM XPCC FUNCTION
         EJECT
*************************************************************************
*        XPCC FUNCTION TERMINATE (LOGOFF)
*************************************************************************
         LA    R5,IJBXTRM              FUNCTION TERMINATE
         BAL   R6,XPCCSERV             PERFORM XPCC FUNCTION
         EJECT
*************************************************************************
*        RETURN CONTROL TO CALLING PROGRAM
*************************************************************************
         L     R13,SAVEAREA+4          CALLING PROGRAM'S SAVE AREA
         RETURN (14,12)
         EJECT
```

```
*********************************************************************
*        PERFORM XPCC FUNCTION
*
*        HOW TO CALL IT :
*        LA    R5,'XPCC FUNCTION BYTE'
*        BAL   R6,XPCCSERV
*********************************************************************
XPCCSERV DS    ØH
         XPCC  XPCCB=(R4),                                          *
               FUNC=(R5)
         EJECT
*********************************************************************
*        TEST XPCC RETURN CODES
*********************************************************************
         LTR   R15,R15                TEST RETURN CODE IN REGISTER
                                       15
         BZR   R6                     NO ERROR
         CLI   IJBXFCT,IJBXCON        FUNCTION CONNECT
         BNE   ERROR                  NO, ERROR
         CLI   IJBXTOAP,X'ØØ'         PARTNER ANY
         BER   R6                     YES, DO NOT WAIT
         CLI   IJBXRETC,IJBXNIDN      PARTNER'S IDENTIFY MISSING
         BE    WAITOTH                YES, WAIT
         CLI   IJBXRETC,IJBXNCNN      PARTNER'S CONNECT STILL
                                       MISSING
         BNE   ERROR                  NO, OTHER ERROR
         EJECT
*********************************************************************
*        WAIT FOR PARTNER
*********************************************************************
WAITOTH  DS    ØH
         LA    R1,IJBXCECB            LOAD ADDRESS OF CONNECT ECB
         WAIT  (1)
         BR    R6                     CONTINUE PROCESSING
         EJECT
*********************************************************************
*        ERROR, SET UP OUTPUT PARAMETER
*********************************************************************
ERROR    DS    ØH
         CLI   XPCCFKT,X'ØØ'          FIRST ERROR
         BNER  R6                     NO, KEEP OLD INFORMATION
         STH   R15,XPCCR15            SAVE REGISTER 15
         MVC   XPCCFKT,IJBXFCT        SAVE XPCC FUNCTION
         MVC   XPCCRC,IJBXRETC        SAVE XPCC RETURN CODE
         BR    R6                     CONTINUE PROCESSING
         EJECT
*********************************************************************
*        SAVE AREA
*********************************************************************
```

```
SAVEAREA DS    9D
         EJECT
*************************************************************************
*        CROSS-PARTITION COMMUNICATION CONTROL BLOCK (MODIFIED)
*************************************************************************
BATXPCCB XPCCB APPL=MYAPLL,                                            *
               TOAPPL=YOURAPPL
         EJECT
*************************************************************************
*        CROSS-PARTITION COMMUNICATION CONTROL BLOCK (CONSTANT)
*************************************************************************
KSTXPCCB XPCCB APPL=MYAPPL,                                            *
               TOAPPL=YOURAPPL
         END
```

Walter Richters
(Germany)

GET/PUT // SETPARM statement subroutine

This subroutine retrieves the value of a previously entered // SETPARM statement and returns it to the calling program, or simulates the setting of a // SETPARM statement just as if it were entered via JCL.

One parameter must be passed, consisting of the following five fields:

- *Function* – A one-byte field indicating the function to be performed. Specify 'P' for PUT or 'G' for GET. If 'P' is not specified, 'G' is assumed.
- *Return code* – A four-byte field which, on return to the calling program, will contain a return code in binary form. The return code is issued by $IJBPROC by simulating the PARMMAC macro function. A non-zero value indicates that some form of error occurred. Consult the *Initial Program Load and Job Control Logic Manual* (LY33-9110) for an explanation.
- *PARM length* – A two-byte field which will contain the length of the parameter, if found and no error occurred, in binary form – that is, a parameter with a value of 'ABCDEFGHIJKLMN' will be returned with a length of X'000E'. If an error occurred, this

field will contain low-values (ie X'00's). This field needn't be cleared or reset before each call.

- *PARM name* – A seven-byte field which must contain the name of the parameter. If the first field specified 'G' (GET), this field must contain the name of the parameter whose value is to be returned. If 'P' (PUT) was specified, this field must contain a name whose value is to be set, and the value must be specified in the next field. If less than seven bytes, it must be left-justified and padded on the right with blanks.
- *PARM value* – A fifty-byte field. If 'G' was specified, then, on return to the calling program, this field will contain the value of the parameter, if found. If the PARM name specified in the preceding field wasn't found, spaces are returned in this field along with an appropriate return code. If 'P' was specified, this field must contain the value you wish to assign to the PARM name specified in the preceding field. If less than fifty bytes, it must be left-justified and padded on the right with spaces.

NOTES

- It is highly recommended that the return code be examined after each call, so that appropriate action can be taken should an error occur.
- Note that null PARM values (ie '// SETPARM PARM=' or '// SETPARM PARM="') are returned with a length value of low-values (ie X'0000') and the PARM value will contain all low-values.
- If you attempt to return a PARM name for which no previous '// SETPARM PARM=??' statement was given, you will get a return code of X'00000010'.
- If a PARM value is set to a specific value and subsequently reset via another call to low-values, an attempt to 'GET' the parameter value will yield a 'Not found' condition.

CALLING SEQUENCES

The calling sequences are given below.

COBOL

```
    CALL 'DPGPPM' USING PARM.
```

ALC

```
    LA    13,SAVEAREA (13 CAN ALSO BE R13 OR RD).
          CALL  DPGPPM,(PARM)
          .
          .  (MAINLINE PART OF PROGRAM).
          .
    SAVEAREA DC    18F'Ø'
```

RPGII

```
    CALL 'DPGPPM'             XX  (XX=ANY UNUSED INDICATOR)
                  PARM           PARM
    SETOF                    XX
```

An eighteen-word save area must be passed through register 13 by the user (standard COBOL linkage).

DPGPPM

```
DPGP     TITLE 'DPGPPM - 1.Ø - GET/PUT // SETPARM STATEMENT
                SUBROUTINE.X'
DPGPPM   CSECT Ø
         USING DPGPPM,RF           ESTABLISH ENTRYPOINT AS BASE REG.
         STM   RE,RC,12(RD)        SVE CALLERS REGS.
         LR    R3,RF               LOAD NEW BASE REG.
         DROP  RF                  DROP TEMPORARY BASE REG.
         USING DPGPPM,R3           ESTABLISH NEW BASE REG.
         LA    RF,SAVEAREA         LOAD ADDRESS OF MY SAVEAREA.
         ST    RF,8(,RD)           PUT IN 3RD WORD OF CALLER'S
                                   SAVEAREA
         ST    RD,4(,RF)           PUT CALLER'S ADDRESS IN MY 2ND
                                   WORD.
         LR    RD,RF               SET SAVEAREA REGISTER TO MY
                                   SAVEAREA
         ST    RD,Ø(,RD)           PUT IN 1ST WORD OF MY SAVEAREA.
         B     GPPMØØ              BRANCH TO GPPMØØ.
*
RØ       EQU   Ø                   EQUATE RØ TO REGISTER ZERO.
R1       EQU   1                   EQUATE R1 TO REGISTER ONE.
```

```
R2       EQU   2                   EQUATE R2 TO REGISTER TWO.
R3       EQU   3                   EQUATE R3 TO REGISTER THREE.
R4       EQU   4                   EQUATE R4 TO REGISTER FOUR.
R5       EQU   5                   EQUATE R5 TO REGISTER FIVE.
R6       EQU   6                   EQUATE R6 TO REGISTER SIX.
R7       EQU   7                   EQUATE R7 TO REGISTER SEVEN.
R8       EQU   8                   EQUATE R8 TO REGISTER EIGHT.
R9       EQU   9                   EQUATE R9 TO REGISTER NINE.
RA       EQU   1Ø                  EQUATE RA TO REGISTER TEN.
RB       EQU   11                  EQUATE RB TO REGISTER ELEVEN.
RC       EQU   12                  EQUATE RC TO REGISTER TWELVE.
RD       EQU   13                  EQUATE RD TO REGISTER THIRTEEN.
RE       EQU   14                  EQUATE RE TO REGISTER FOURTEEN.
RF       EQU   15                  EQUATE RF TO REGISTER FIFTEEN.
*
         DC    C'DPGPPM STARTS HERE. ' INSERT EYE CATCHER.
*
GPPMØØ   EQU   *
         L     R4,Ø(R1)            LOAD ADDRESS OF PASSED PARAMETER.
         CLI   Ø(R4),C'P'          IS FUNCTION 'P' (PUT).
         BE    PPPMØØ              YES-BRANCH TO PPPMØØ
         CLI   Ø(R4),C'S'          IS FUNCTION 'S' (SET).
         BE    PPPMØØ              YES-BRANCH TO PPPMØØ
*        GETSYMB AREA=AREA,PARMNAM=PARMNAME,VALBUF=PARMVALU,
               LENFLD=PARMLENG
*        ST    RF,PARMRTNC
         MVC   PARM,Ø(R4)          MVE PASSED PARAMETER FIELDS.
         MVC   PARMLENG,=H'5Ø'     SET PARM LENGTH TO DEFAULT.
         MVI   PARMVALU,C' '       SET PARM VALUE TO BLANKS.
         MVC   PARMVALU+1(L'PARMVALU-1),PARMVALU ...
         LOAD  $IJBPROC            GET ENTRY ADDRESS OF $IJBPROC.
         LR    RF,R1               LOAD ENTRY ADDRESS TO REG 15.
         LA    R1,CTRL             LOAD ADDRESS OF CTRL TO REG 1.
         LA    RD,SAVEAREA         LOAD ADDRESS OF SAVEAREA TO REG
                                   13.
         BALR  RE,RF               INVOKE $IJBPROC.
         ST    RF,1(R4)            SVE RETURN CODE.
         LTR   RF,RF               WAS THERE AN ERROR.
         BZ    GPPM1Ø              NO-BRANCH TO GPPM1Ø.
         MVC   5(L'PARMLENG,R4),=H'Ø' SET LENGTH OF PARM VALUE TO
                                   ZERO.
         MVI   14(R4),C' '         SET PARM VALUE TO BLANKS.
         MVC   15(L'PARMVALU-1,R4),14(R4)    ...
         B     GPPM3Ø              BRANCH TO GPPM3Ø.
*
GPPM1Ø   EQU   *
         MVC   5(L'PARMLENG,R4),PARMLENG MVE LENGTH OF PARM VALUE TO
                                   RE
         SR    R6,R6               CLEAR REG 6.
         ICM   R6,B'ØØ11',PARMLENG LOAD LENGTH OF PARM VALUE TO REG
                                   6.
```

```
         BZ    GPPM3Ø              ZERO-BRANCH TO GPPM3Ø.
         BCTR  R6,Ø                MAKE IT MVC LENGTH.
         EX    R6,GPPM5Ø           MVE PARM VALUE.
*
GPPM3Ø   EQU   *
         L     RD,4(,RD)           LOAD CALLERS SAVEAREA ADDRESS TO
                                   REG
         ST    RF,16(,RD)          STORE RETURN CODE IN CALLERS
                                   SAVEARE
         LM    RE,RC,12(RD)        RESTORE CALLERS REG.
         BR    RE                  RETURN TO CALLER.
*
GPPM5Ø   MVC   14(1,R4),PARMVALU   MVE RETURNED VALUE.
*
PPPMØØ   EQU   *
         XC    K1BUF,K1BUF         CLEAR BUFFER.
         MVC   5(L'PARMLENG,R4),=H'5Ø' SET PARM LENGTH TO DEFAULT.
         MVC   PPPMTABL(2),=H'5Ø'  ...
         MVC   PPPMLENG+2(2),=H'5Ø' ...
         MVC   PPPMNAME,7(R4)      MVE PARM NAME.
         MVC   PPPMVALU,14(R4)     MVE PARM VALUE.
         LA    R8,PPPMVALU+L'PPPMVALU-1 LOAD BACK END OF PARM VALUE
                                   TO
         LA    R9,L'PPPMVALU       LOAD LENGTH OF PARM VALUE TO REG
                                   9.
*
PPPM1Ø   EQU   *
         CLI   Ø(R8),C' '          IS THIS POSITION A BLANK.
         BNE   PPPM2Ø              NO-BRANCH TO PPPM2Ø.
         BCTR  R8,Ø                BACK UP ONE (1) POSITION.
         BCT   R9,PPPM1Ø           BRANCH TO PPPM1Ø UNTIL REG 9 ZERO.
         B     PPPM3Ø              BRANCH TO PPPM3Ø.
*
PPPM2Ø   EQU   *
         ST    R9,SVLENG           SVE LENGTH OF PARM VALUE.
         STCM  R9,B'ØØ11',5(R4)    STORE IT IN PASSED PARAMETER.
         MVC   PPPMLENG+2(2),SVLENG+2 MVE IT FOR $IJBPROC.
         MVC   PPPMTABL(2),SVLENG+2 ...
         MVC   1(2,R8),=X'FFFF'    MVE PARM VALUE END INDICATOR.
*
PPPM3Ø   EQU   *
         LOAD  $IJBPROC            LOAD $IJBPROC.
         LR    RF,R1               LOAD ENTRY POINT TO REG 15.
         MVI   K1BUF,8             INDICATE SETPAR FUNCTION.
         LA    R1,K1BUF            LOAD ADDRESS OF BUFFER TO REG 1.
         MVI   PPPMTABL+2,X'8Ø'    INDICATE PARAMETER VALUE NOT
                                   NULLIFI
         CLC   PPPMVALU,LOWVALU    IS PASSED PARM VALUE LOW-VALUES.
         BH    PPPM4Ø              NO-BRANCH TO PPPM4Ø.
         MVI   PPPMTABL+2,X'4Ø'    INDICATE PARAMETER VALUE
                                   NULLIFIED.
```

```
         MVC   5(2,R4),=H'Ø'       SET LENGTH TO ZERO.
         MVC   PPPMTABL(2),=H'Ø'   ...
         MVC   PPPMLENG+2(2),=H'Ø' ...
*
PPPM4Ø   EQU   *
         LA    RØ,PPPMTABL         LOAD ADDRESS OF TABLE TO REG Ø.
         ST    RØ,16(R1)           STORE IT IN 5TH FULL WORD.
         SR    RØ,RØ               CLEAR REG Ø.
         LA    RD,SAVEAREA         LOAD ADDRESS OF SAVEAREA FOR
                                   $IJBPRO
         BALR  RE,RF               INVOKE $IJBPROC.
         ST    RF,1(R4)            SVE RETURN CODE.
         B     GPPM3Ø              BRANCH TO GPPM3Ø.
*
         LTORG                     DISPLAY LITERALS.
*
DPGPPMS  DC    C'DPGPPM STORAGE HERE. ' INSERT EYE CATCHER.
*
         DS    ØD                  FOR ALIGNMENT.
*REA     DS    CL1ØØ
CTRL     DC    X'Ø1ØØØØØØ',AL4(PARMNAME),AL4(PARMVALU),AL4(PARMLENG)
PARM     DS    ØCL64
PARMFUNC DS    C                   FUNCTION.
PARMRTNC DS    CL4                 RETURN CODE.
PARMLENG DS    CL2                 RETURN PARM VALUE LENGTH.
PARMNAME DS    CL7                 NAME OF PARM.
PARMVALU DS    CL5Ø                RETURN PARM VALUE.
*
         DS    ØD                  FOR ALIGNMENT.
K1BUF    DS    CL24                BUFFER USED FOR PASSING PARM TO
                                   $IJB
PPPMTABL DC    H'Ø',X'ØØ'          FORMATTED TABLE FOR PASSING TO
                                   $IJBP
PPPMNAME DC    CL7' '
PPPMVALU DC    CL5Ø' '
         DC    X'FFFF'             END OF TABLE. (DON'T REMOVE).
PPPMLENG DC    F'Ø'
*
LOWVALU  DC    XL5Ø'ØØ'
*
SVLENG   DS    F
SAVERD   DS    F                   SVE AREA FOR REG 13.
SAVEAREA DS    18F                 SVE AREA FOR $IJBPROC.
*
DPGPPME  DC    X'FF'
*
         END
```

Robert Botsis
Senior Systems Programmer (USA)

Automated compile generator

My company develops applications of various types, including COBOL, Assembler, Maps, and AFP resources. We also use SQL, and support CICS and batch environments. In order to allow our programmers to concentrate on coding rather than on maintaining compile JCL, we developed the PGM system presented here to automatically generate the required JCL and submit their programs for compilation.

PGM runs in CMS (we use XEDIT for development), and displays a screen (shown below) so that the user can make selections to describe the type of compile he/she wants. The PGM2 EXEC then generates and submits the required JCL. During this process, an auxiliary file is written to the user's A-disk to save his panel settings for future compiles so that he doesn't have to re-enter the selections. We use CA-Panvalet as our source manager, so the initial compile step updates Panvalet to keep the source code current. For efficiency, successive steps are punched to POWER's internal reader, and start executing immediately after the previous step finishes.

PGM uses a locally written program called VSICP to perform minor variable substitution and submit the JCL to VSE after including the required JCL procs which reside as separate members in CMS. However, VSICP can be replaced by adding the JCL procs to the end of the PGM2 EXEC as called routines, prefixing each line with PUSH statements, and using CP SPOOL PUNCH to route and submit the job (we are considering doing this). The JCL procs use a program called $SYSPCH to punch successive steps to the internal reader, but this can be replaced with IBM's IESINSRT which does much the same thing.

The PGM main menu is shown below.

```
*************************************************************************
***  PGM Main Panel                                                   ***
*************************************************************************

  PGMP00                  Program Generation Menu           09/10/1997

  File Name ==>           PGM0002  Auxfile not specified or not found.
```

```
       Type ==> COBOL    Default settings applied.
       Mode ==> A

Program Type ==> CCMD
 CCMD  Cob CICS          ACMD  Asm CICS           DSECT map (DSECT)
 CCMDC Cob CICS called   ABAT  Asm batch          MAP   map (physical
 CBAT  Cob batch         ACALL Asm batch called   DYLAK Dylakor
                                                    freeze
 CCALL Cob batch called     ASQL  Asm CICS SQL       FDEF  Formdef
 CSQL  Cob CICS SQL         ASQLB Asm batch SQL      PDEF  Pagedef
 CSQLC Cob CICS SQL called                           OVLY  Overlay
 CSQLB Cob batch SQL
 CSQBC Cob batch SQL called

 Phase Name     ===>     NOLINK to skip lnkedt;  Default is File Name
 Panvalet Lib   ===> 1  1 Production 2 Programmer 3 Data Adm.
 VSE Sublib     ===> 1  1 LIB1.TEST  2 LIB1.TEST2
 Listing Wanted ===> 4  1 Panvalet   2 SQL prep   3 CICS prep  4 Compile
 Debugger Type  ===> 1  1 None       2 Intertest  3 Simon
 Cobol Type     ===> 2  1 Non-Cobol  2 Cobol/VSE  3 Cobol II  4 DOS/
                                                              Cobol
 Enter Submit Job   PF3 Exit
```

PGM has been in use for several years at our site under many different combinations of VM and VSE with no problems.

PGM EXEC

```
*************************************************************************
***  PGM EXEC...                                                      ***
*************************************************************************

/************************************************************************/
/************************************************************************/
/***                                                                  ***/
/***  Program Name  - PGM                                             ***/
/***  Installation  - Arkansas Farm Bureau Insurance Company          ***/
/***  Author        - Technical Support                               ***/
/***  Date Written  - Ø1/19/9Ø, re-written Ø8/3Ø/97                   ***/
/***                                                                  ***/
/***  Function   Display an ISPF panel allowing users to select       ***/
/***             from various types of compiles such as maps,         ***/
/***             programs, AFP resource gens ...etc.  Then build      ***/
/***             and submit the JCL to perform the compilation.       ***/
/***                                                                  ***/
/***  Inputs     PGM can be invoked from the command line with        ***/
/***             or without an auxfile specification.                 ***/
/***             Auxfiles are stored on the A-disk during             ***/
```

```
/***             each compile and contain settings that were       ***/
/***             selected during the last compile.  For example,   ***/
/***             entering                                          ***/
/***                                                               ***/
/***                PGM         shows screen w/default settings    ***/
/***                                                               ***/
/***                PGM AUT28I  shows screen w/settings used       ***/
/***                            during last compile of AUT28I      ***/
/***                                                               ***/
/***             PGM can also be invoked from FILELIST to pick     ***/
/***             up on the FILENAME, FILETYPE and FILEMODE of      ***/
/***             any entry.  For example:                          ***/
/***                                                               ***/
/***                 CM24ØØØ9 FILELIST AØ  V 1Ø8  Trunc=1Ø8 S      ***/
/***                Cmd   Filename Filetype Fm Format Lrecl        ***/
/***                      ISPSPROF ISPPROF  A1 V       1719        ***/
/***                      PGMPROF  ISPPROF  A1 V       1335        ***/
/***                      SGB15A   ASSEMBLE A1 V         71        ***/
/***                      PGX      XEDIT    A1 V         13        ***/
/***                      SGBØ9I   AUXFILE  A1 V         87        ***/
/***                ==> PGM   SGBØ9I   COBOL    A1 V         8Ø        ***/
/***                      PGX      EXEC     A1 V         45        ***/
/***                      QDASD    ACTIVITY AØ V         41        ***/
/***                                                               ***/
/***  Output     JCL member is built and submitted to VSE by an    ***/
/***             assembler submit program (VSICP) which performs   ***/
/***             variable substitution and other functions.        ***/
/***                                                               ***/
/*********************************************************************/
/*********************************************************************/

   PARSE UPPER ARG JNAME JTYPE JMODE .

  'FILEDEF ISPPLIB DISK ISPNULL PANEL * (PERM CONCAT)'
  'FILEDEF ISPMLIB DISK ISPNULL MESSAGE * (PERM CONCAT)'

  'EXEC ISPSTART CMD(PGM2' JNAME JTYPE JMODE') NEWAPPL(PGM)'

  'FILEDEF ISPPLIB CLEAR'
  'FILEDEF ISPMLIB CLEAR'

  'CP SET IMSG ON'  ;
  'CP SET EMSG ON'


*********************************************************************
***  PGM2.EXEC which is driven by PGM.EXEC...                     ***
*********************************************************************
```

```
/*********************************************************************/
/*********************************************************************/
/***                                                               ***/
/***  Program Name  - PGM2                                         ***/
/***  Installation  - Arkansas Farm Bureau Insurance Company       ***/
/***  Author        - Technical Support                            ***/
/***  Date Written  - Ø1/19/9Ø, re-written Ø8/3Ø/97                ***/
/***                                                               ***/
/***  Function:  Extension of the PGM exec.  PGM2 performs the     ***/
/***             calculations, JCL generation ...etc necessary     ***/
/***             to perform the requested compilation.             ***/
/***                                                               ***/
/*********************************************************************/
/*********************************************************************/

   PARSE UPPER ARG JNAME JTYPE JMODE .

   ZPFCTL = 'OFF'                      /* NO PF KEYS ON ISPF PANEL     */
  'SET IMSG OFF'  ;  'SET EMSG OFF'    /* TURN OFF MESSAGES            */
   ADDRESS ISPEXEC 'VPUT (ZPFØ1 ZPFØ2 ZPFØ3 ZPFØ4 ZPFØ5 ZPFØ6 ZPFØ7
    ZPFØ8 ZPFØ9 ZPF1Ø ZPF11 ZPF12) PROFILE'
   ADDRESS ISPEXEC 'VPUT (ZPF13 ZPF14 ZPF15 ZPF16 ZPF17 ZPF18 ZPF19
    ZPF2Ø ZPF21 ZPF22 ZPF23 ZPF24) PROFILE'
   ADDRESS ISPEXEC 'VPUT (ZPFCTL) PROFILE'

/*********************************************************************/
/***                                                               ***/
/***  PROGRAM-TYPE TABLE & RELATED VARIABLES...                    ***/
/***                                                               ***/
/***  P. = PROGRAM TYPE              S. = POWER USER-INFO DATA     ***/
/***  I. = INCLUDE-MEMBER NAME       V. = PANVALET SOURCE TYPE     ***/
/***                                                               ***/
/*********************************************************************/
   INIT:

P.    = ' '    ; I.    = ' '       ; S.    = ' '      ; V.    = ' '
P.1   = 'CCMD' ; I.1   = 'PGMCCMD' ; S.1   = 'CICS'   ; V.1   = 'COBOL'
P.2   = 'CCMDC'; I.2   = 'PGMCCMC' ; S.2   = 'CICS'   ; V.2   = 'COBOL'
P.3   = 'CBAT' ; I.3   = 'PGMCBAT' ; S.3   = 'COMPILE'; V.3   = 'COBOL'
P.4   = 'CCALL'; I.4   = 'PGMCCALL'; S.4   = 'COMPILE'; V.4   = 'COBOL'
P.5   = 'CSQL' ; I.5   = 'PGMCSQL' ; S.5   = 'SQL'    ; V.5   = 'COBOL'
P.6   = 'CSQLC'; I.6   = 'PGMCSQLC'; S.6   = 'SQL'    ; V.6   = 'COBOL'
P.7   = 'CSQLB'; I.7   = 'PGMCSQLB'; S.7   = 'SQL'    ; V.7   = 'COBOL'
P.8   = 'CSQBC'; I.8   = 'PGMCSQBC'; S.8   = 'SQL'    ; V.8   = 'COBOL'
P.9   = 'ACMD' ; I.9   = 'PGMACMD' ; S.9   = 'CICS'   ; V.9   = 'BAL'
P.1Ø  = 'ABAT' ; I.1Ø  = 'PGMABAT' ; S.1Ø  = 'COMPILE'; V.1Ø  = 'BAL'
P.11  = 'ACALL'; I.11  = 'PGMACALL'; S.11  = 'COMPILE'; V.11  = 'BAL'
P.12  = 'ASQL' ; I.12  = 'PGMASQL' ; S.12  = 'SQL'    ; V.12  = 'BAL'
P.13  = 'ASQLB'; I.13  = 'PGMASQLB'; S.13  = 'SQL'    ; V.13  = 'BAL'
```

```
P.14  = 'DSECT'; I.14  = 'PGMADSCT'; S.14  = 'DSECT'  ; V.14  = 'OTHER'
P.15  = 'MAP'  ; I.15  = 'PGMAMAP' ; S.15  = 'COMPILE'; V.15  = 'OTHER'
P.16  = 'DYLAK'; I.16  = 'PGMDYLAK'; S.16  = 'FRZLINK'; V.16  = 'DYLAK'
P.17  = 'FDEF' ; I.17  = 'PGMXDEF' ; S.17  = 'AFPGEN' ; V.17  = 'OTHER'
P.18  = 'PDEF' ; I.18  = 'PGMXDEF' ; S.18  = 'AFPGEN' ; V.18  = 'OTHER'
P.19  = 'OVLY' ; I.19  = 'PGMOVLY' ; S.19  = 'AFPGEN' ; V.19  = 'OTHER'

/*********************************************************************/
/***  GENERAL DEFAULTS...                                          ***/
/*********************************************************************/

 CLASS2   = 'C'                       /* COMPILE CLASS               */
 DEBUGØ   = "'.*'"                    /* DBGGR FORMATTED DUMP OPT    */
 DEBUG1   = "'.*'"                    /* DBGGR PWR LST STMT          */
 DEBUG2   = "'.*'"                    /* DBGGR PWR LST STMT CONT     */
 DEBUG3   = "'.*'"                    /* DBGGR EXEC STMT             */
 DEBUG4   = "'.*'"                    /* DBGGR PARM STMT             */
 LSTCICS  = 'SOURCE'                  /* CICS-PREP PRINT OPT         */
 SQLUSER  = 'SQL'                     /* SQL-PREP USERID             */
 SQLPWD   = 'SQL141T'                 /* SQL-PREP PASSWORD           */
 USERID   = USERID()                  /* CMS USERID                  */
 WDATE    = SUBSTR(DATE(U),1,6)       /* DATE TO DISPLAY IN          */
 WDATE    = WDATE || WORD(DATE(N),3)  /*   INITIAL PANEL             */

/*********************************************************************/
/***  RESTORE COMPILE OPTIONS FROM AUXFILE IF PRESENT...           ***/
/*********************************************************************/

 IF JNAME = '' THEN DO
    CALL DEFAULT
    SIGNAL NOAUX  ;  END

 JMODE = SUBSTR(JMODE,1,1)            /* ALLOW FOR A1, A2 ...ETC     */
   ADDRESS COMMAND 'LISTFILE' JNAME 'AUXFILE' JMODE '(STACK DATE'

   IF RC = Ø THEN DO
      PARSE PULL ENTRY
      PARSE VAR ENTRY . . . . . . . CDATE CTIME .
      MSG1 = 'PGMØØØ1  Parms applied from existing auxfile.'
      MSG2 = 'Last compiled' CDATE 'at' CTIME
     'EXECIO 13 DISKR' JNAME 'AUXFILE' JMODE '(FINIS'
      PARSE UPPER PULL .                        /* COMMENT CARD    */
      PARSE UPPER PULL . JNAME
      PARSE UPPER PULL . JTYPE
      PARSE UPPER PULL . JMODE
      PARSE UPPER PULL . JPTYP
      PARSE UPPER PULL . JPHASE
      PARSE UPPER PULL . JPANLIB
      PARSE UPPER PULL . JVSE
      PARSE UPPER PULL . JLISTW
```

```
      PARSE UPPER PULL . JLISTD
      PARSE UPPER PULL . JDEBUG
      PARSE UPPER PULL . JCTYP
      PARSE UPPER PULL . JCTYPXX       /* ONLY IN OLD AUXFILES        */
      CALL CONVERT  ;  END
   ELSE DO
      CALL DEFAULT  ;  END

   NOAUX:

   CALL DISP00

   CKRESP:

   IF CPFKEY = 'PF03' THEN EXIT 4

   IF CPFKEY ¬= ' ' THEN DO
      MSG1 = 'PGM0003  Invalid PF key selected.'
      CALL DISP00  ;  SIGNAL CKRESP  ;  END

/*********************************************************************/
/***  VALIDATE FILENAME...                                         ***/
/*********************************************************************/

   IF JNAME = '' | JTYPE = '' | JMODE = '' THEN DO
      MSG1 = 'PGM0004  File information must be specified.'
      CALL DISP00  ;  SIGNAL CKRESP  ;  END

  'STATE' JNAME JTYPE JMODE

   IF RC > '0' THEN DO
      MSG1 = 'PGM0005  Unable to locate specified file.'
      MSG2 = '         Verify file information is correct.'
      CALL DISP00  ;  SIGNAL CKRESP  ;  END
   ELSE DO
      TJNAME = SUBSTR(JNAME,1,7)
      TJNAME = STRIP(TJNAME,B,' ')  ;  END

/*********************************************************************/
/***  VALIDATE PROGRAM TYPE...                                     ***/
/*********************************************************************/

   NDX = 1
   DO UNTIL P.NDX = ' ' | NDX = 40       /* 40 = LOOP PREVENTER       */
      IF JPTYP = P.NDX THEN LEAVE
      NDX = NDX + 1  ;  END

   IF P.NDX = ' ' THEN DO
      MSG1 = 'PGM0006  Program type is invalid.  Choose a'
```

```
      MSG2 = '         program type from the list.     '
      CALL DISPØØ  ;  SIGNAL CKRESP  ;  END

   IF JCTYP > 1 THEN DO
      IF SUBSTR(JPTYP,1,1) = 'C' THEN DO
         NOP ; END
      ELSE DO
         MSG1 = 'PGMØØ21  Program type invalid for type of  '
         MSG2 = '         Cobol selected.                 '
         CALL DISPØØ  ;  SIGNAL CKRESP  ;  END  ;  END
   ELSE DO
      IF SUBSTR(JPTYP,1,1) = 'C' THEN DO
         MSG1 = 'PGMØØ21  Program type invalid for type of  '
         MSG2 = '         Cobol selected.                 '
         CALL DISPØØ  ;  SIGNAL CKRESP  ;  END
      ELSE DO
         NOP  ;  END  ;  END

   INCTYPE = I.NDX  ;  STEPD = S.NDX  ;  PVTYPE = V.NDX

/*********************************************************************/
/***  VALIDATE PHASE NAME...                                       ***/
/*********************************************************************/

   IF JPHASE = '' THEN JPHASE = JNAME

   IF JPANLIB < 1 | JPANLIB > 3 THEN DO
      MSG1 = 'PGMØØØ7  Invalid Panvalet library specified.'
      CALL DISPØØ  ;  SIGNAL CKRESP  ;  END

   IF JVSE = '1' THEN
      SUBLIB = 'LIB1.TEST'
   ELSE
      IF JVSE = '2' THEN
         SUBLIB = 'LIB1.TEST2'
      ELSE DO
         MSG1 = 'PGMØØØ8  Invalid VSE sublib specified.'
         CALL DISPØØ  ;  SIGNAL CKRESP  ;  END

   IF JLISTW < 1 | JLISTW > 4 THEN DO
      MSG1 = 'PGMØØØ9  Invalid listing-wanted specification.'
      MSG2 = '         Choose the listings you want generated.'
      CALL DISPØØ  ;  SIGNAL CKRESP  ;  END

/*********************************************************************/
/***  VALIDATE LISTING DESTINATION OPTION...                       ***/
/*********************************************************************/

   IF JLISTD = 1 THEN DO
      LDISP = 'D'  ;  LCLASS = 'R'  ;  LDEST = USERID()
```

```
      LDISP1= 'D'  ;  LCLASS1= 'R'  ;  LDEST1= USERID()
      LDISP2= 'D'  ;  LCLASS2= 'R'  ;  LDEST2= USERID()
      LDISP3= 'D'  ;  LCLASS3= 'R'  ;  LDEST3= USERID()
      LDISP4= 'D'  ;  LCLASS4= 'R'  ;  LDEST4= USERID() ; END
   ELSE
      IF JLISTD = 2 THEN DO
         LDISP = 'H'  ;  LCLASS = 'S'  ;  LDEST = 'IS06'
         LDISP1= 'H'  ;  LCLASS1= 'S'  ;  LDEST1= 'IS06'
         LDISP2= 'H'  ;  LCLASS2= 'S'  ;  LDEST2= 'IS06'
         LDISP3= 'H'  ;  LCLASS3= 'S'  ;  LDEST3= 'IS06'
         LDISP4= 'H'  ;  LCLASS4= 'S'  ;  LDEST4= 'IS06'  ;  END
      ELSE DO
         MSG1 = 'PGM0010  Invalid listing destination specified.'
         CALL DISP00  ;  SIGNAL CKRESP  ;  END

/*********************************************************************/
/***  VALIDATE DEBUGGER OPTION...                                  ***/
/*********************************************************************/

   IF JDEBUG < 1 | JDEBUG > 3 THEN DO
      MSG1 = 'PGM0011  Invalid debugger selection specified.'
      CALL DISP00  ;  SIGNAL CKRESP  ;  END

   IF JDEBUG = '1' THEN SIGNAL ENDDBUG

   IF JDEBUG = '3' & JCTYP = '1' THEN DO
      MSG1 = 'PGM0037  Simon support not yet implemented for'
      MSG2 = 'this program type. Select again.'
      CALL DISP00  ;  SIGNAL CKRESP  ;  END

   IF JDEBUG = '2' & (JPTYP = 'CCMD' | JPTYP = 'CCMDC' | ,
      JPTYP = 'CSQL' | JPTYP = 'CSQLC' | JPTYP = 'ACMD' | ,
      JPTYP = 'ASQL') THEN
        SIGNAL ENDDBUG

   IF JDEBUG = '3' & (JPTYP = 'CBAT' | JPTYP = 'CCALL' | ,
      JPTYP = 'CSQLB' | JPTYP = 'CSQBC' | JPTYP = 'ABAT' | ,
      JPTYP = 'ACALL' | JPTYP = 'ASQLB') THEN
        SIGNAL ENDDBUG

   MSG1 = 'PGM0012  Wrong debugger for this type of program.'
   MSG2 = '         Verify debugger and program type selections.'
   CALL DISP00  ;  SIGNAL CKRESP

   ENDDBUG:

/*********************************************************************/
/***  VALIDATE COBOL TYPE...                                       ***/
/*********************************************************************/
```

```
   IF JCTYP < 1 | JCTYP > 4 THEN DO
      MSG1 = 'PGMØØ13  Invalid Cobol type selected.'
      CALL DISPØØ  ;  SIGNAL CKRESP  ;  END

   IF JCTYP = '2' THEN DO
      LIBDEF1 = "'// LIBDEF
*,SEARCH=("SUBLIB",PRD2.SCEEBASE,PRD2.PROD,PRD2.DBASE)'"
      LIBDEF2 = "'// LIBDEF *,CATALOG="SUBLIB"'"  ;  END
   ELSE DO
      LIBDEF1 = "'// LIBDEF *,SEARCH="SUBLIB",CATALOG="SUBLIB"'"
      LIBDEF2 = "'.*'"  ;  END

/*********************************************************************/
/***  INPUT PARMS ARE ACCEPTABLE; BUILD JCL VARIABLES...           ***/
/*********************************************************************/

  /* COMMAND  = "'"JNAME JTYPE JMODE JPTYP JPHASE"'"                  */
  /* COMMAND2 = "'"JPANLIB JVSE JLISTW JLISTD JDEBUG JCTYP"'"         */

   IF JCTYP = '1' THEN DO
      EXEC = "'// EXEC ASSEMBLY,SIZE=256K'"
      SIGNAL ENDCOBOL  ;  END

   IF JCTYP = '4' THEN DO
      EXEC = "'// EXEC FCOBOL,SIZE=256K'"

      IF JPTYP = 'CCALL' | JPTYP = 'CSQBC' | JPTYP = 'CCMDC' | ,
         JPTYP = 'CSQLC' THEN
           COMPOPT = "'// OPTION DECK'"
      ELSE
           COMPOPT = "'// OPTION LISTX,SYM,ERRS,CATAL,NODECK'"

      SIGNAL ENDCOBOL  ;  END

/*********************************************************************/
/***  MUST BE COBOL II OR COBOL/VSE FROM THIS POINT FORWARD        ***/
/*********************************************************************/

   EXEC = "'// EXEC IGYCRCTL'"

   IF LSTCICS = 'SOURCE' THEN
      LSTCICS = "'"SOURCE COBOL2"'"
   ELSE
      LSTCICS = 'COBOL2'

   IF JPTYP = 'CCALL' | JPTYP = 'CSQBC' | JPTYP = 'CCMDC' | ,
      JPTYP = 'CSQLC' THEN
        COMPOPT = "'// OPTION DECK'"
   ELSE
```

```
        COMPOPT = "'// OPTION CATAL'"

   IF JPTYP = 'ASQL' | JPTYP = 'ASQLB' | JPTYP = 'CSQBC' |,
      JPTYP = 'CSQL' | JPTYP = 'CSQLB' | JPTYP = 'CSQLC' THEN
        SQLPARM = "'PRINT,COB2'"
   ELSE
        SQLPARM = "'PRINT'"

   ENDCOBOL:

/*********************************************************************/
/***  FOR ALL COMPILES EXCEPT DSECTS, A PANVALET UPDATE JOB IS      ***/
/***  THE FIRST STEP TO RUN.  DSECTS ARE NOT STORED ON PANVALET     ***/
/***  AND DON'T NEED THIS STEP.  THEREFORE, THE FOLLOWING           ***/
/***  SECTION SETS VARIABLES WHICH WILL MODIFY COMMENTS,            ***/
/***  SUFFIXES ETC IN THE PANVALET UPDATE JOB TO REFLECT A          ***/
/***  DSECT AS NECESSARY.                                           ***/
/*********************************************************************/

 IF JPTYP = 'DSECT' THEN DO
    COMSTEP = "'*** PGM - DSECT ASSEMBLY *** '" /* JOBCARD COMMENT    */
    JS      = 'D'                               /* JNAME SUFFIX       */
    STEPD2  = 'DSECT'                           /* POWER USER-INFO    */
    DSECTL1 = "'// GOTO DSECT'"                 /* IF NOT DSECT       */
    DSECTL2 = "'/. DSECT'"                      /* IF NOT DSECT       */
    DSECTL3 = "'/'"  ;  END                     /* IF NOT DSECT       */
   ELSE DO
      COMSTEP = "'*** PGM - PANVALET UPDATE ***'"
      JS      = 'C'
      STEPD2  = 'PANV'
      DSECTL1 = "'.*'"
      DSECTL2 = "'.*'"
      DSECTL3 = '$'    ;  END

/*********************************************************************/
/***  EDIT AND SET LNKEDT PARM...                                   ***/
/*********************************************************************/

   IF JPHASE = 'NOLINK' | JPTYP = 'CCALL' | JPTYP = 'ACALL' | ,
       JPTYP = 'CCMDC'  | JPTYP = 'CSQBC' | JPTYP = 'CSQLC' THEN
        LINK = "'.*'"
   ELSE
      IF JCTYP = '1' | JCTYP = '4' THEN
         LINK = '"// EXEC LNKEDT,PARM=''AMODE=24,RMODE=24''"'
      ELSE
         IF JPTYP = 'CCMD' | JPTYP = 'CSQL' THEN
            LINK  = '"// EXEC LNKEDT,PARM=''AMODE=31,RMODE=ANY''"'
         ELSE
            LINK = '"// EXEC LNKEDT,PARM=''AMODE=24,RMODE=24''"'
```

```
/*********************************************************************/
/***  EDIT AND SET LISTINGS OPTION.  THE SELECTED LISTING AND       ***/
/***  ALL LOWER-LEVEL LISTINGS ARE GENERATED.                       ***/
/*********************************************************************/

   IF JLISTW = '4' THEN DO
      LCLASS1 = 'R'    ;  LCLASS2 = 'R'    ;  LCLASS3 = 'R'
      LDEST1  = 'CSQ'  ;  LDEST2  = 'CSQ'  ;  LDEST3  = 'CSQ'
      LDISP1  = 'D'    ;  LDISP2  = 'D'    ;  LDISP3  = 'D'
      END

   IF JLISTW = '3' THEN DO
      LCLASS1 = 'R'    ;  LCLASS2 = 'R'
      LDEST1  = 'CSQ'  ;  LDEST2  = 'CSQ'
      LDISP1  = 'D'    ;  LDISP2  = 'D'
      END

   IF JLISTW = '2' THEN DO
      LCLASS1 = 'R'
      LDEST1  = 'CSQ'
      LDISP1  = 'D'
      END

/*********************************************************************/
/***  SET JCL COMMENT VARIABLE...                                   ***/
/*********************************************************************/

/*  IF JCTYP = '2' THEN                                              */
/*     COM1 = 'COBOL FOR VSE'                                        */
/*  ELSE                                                             */
/*     IF JCTYP = '3' THEN                                           */
/*        COM1 = 'COBOL II'                                          */
/*    ELSE                                                           */
/*       IF JCTYP = '4' THEN                                         */
/*          COM1 = 'DOS/VS COBOL'                                    */
/*       ELSE                                                        */
/*        IF JPTYP = 'FDEF' | JPTYP = 'PDEF' | JPTYP = 'OVLY' THEN   */
/*             COM1 = 'AFP RESOURCE GENERATION'                      */
/*          ELSE                                                     */
/*             IF JPTYP = 'DYLAK' THEN                               */
/*                COM1 = 'DYLAKOR FREEZE'                            */
/*             ELSE                                                 */
/*                COM1 = 'PROGRAM GENERATION'                       */
/*                                                                  */
/*  TYPECOM = "'"COM1"'"                                            */

/*********************************************************************/
/* SELECTED COBOL II AND COBOL/VSE ONLINE PROGRAMS ARE SET TO RUN    */
/* IN DYNAMIC PARTITIONS (CLASS=J).  EVERYTHING ELSE WILL RUN IN     */
```

```
/* STATIC PARTITIONS (CLASS=C).  FOR COMPILES WITH SQL, THE SQL       */
/* PREP STEPS ARE ALL SINGLE-THREADED THROUGH ONE PARTITION           */
/* (CLASS=B) TO PREVENT SQL DATABASE LOCKOUTS.  AFTER THE PREP        */
/* STEP IS FINISHED, COMPILES FAN BACK OUT INTO MULTIPLE              */
/* PARTITIONS (CLASS=C OR CLASS=J).                                   */
/*********************************************************************/

   IF JCTYP ¬= '2' & JCTYP ¬= '3' THEN
      CLASS2 = 'C'
   ELSE
      IF JPTYP = 'CCMD'  | JPTYP = 'CCMDC' | JPTYP = 'CSQL' | ,
         JPTYP = 'CSQLC' THEN
           CLASS2 = 'J'
      ELSE
           CLASS2 = 'C'

   IF JPTYP = 'ASQL' | JPTYP = 'ASQLB' | JPTYP = 'CSQBC' |,
      JPTYP = 'CSQL' | JPTYP = 'CSQLB' | JPTYP = 'CSQLC' THEN
      CLASS = 'B'
   ELSE
      CLASS = CLASS2

/*********************************************************************/
/***  SET ADDITIONAL DEBUGGER OPTIONS...                           ***/
/*********************************************************************/

 IF JDEBUG = '1' THEN                /* IF NO DEBUG, USE DEFAULTS    */
    NOP
 ELSE
    IF JDEBUG = '2' THEN             /* SET FOR INTERTEST            */
       CALL INTRTEST
    ELSE
       CALL SIMON                    /* SET FOR SIMON                */

 IF JCTYP = '1' | JCTYP = '4' THEN SIGNAL ENDDBUG2

 IF JPTYP ¬= 'CBAT' & JPTYP ¬= 'CCALL' ,
    & JPTYP ¬= 'CSQLB' & JPTYP ¬= 'CSQBC' THEN
      SIGNAL ENDDBUG2

 IF JCTYP = '2' THEN
    DEBUGØ = "' CBL TEST(NONE,SYM)'"       /* SET FOR COBOL/VSE       */
 ELSE
    DEBUGØ = "' CBL FDUMP'"                /* SET FOR COBOL II        */

 ENDDBUG2:

/*********************************************************************/
/***  BUILD AUXFILE WITH PGM VARIABLES AND WRITE TO USERS A-DISK    ***/
/*********************************************************************/
```

```
 'ERASE' JNAME 'AUXFILE A1'  ;  'DESBUF'

  PUSH '.SET  VAR=CLASS         DATA='CLASS
  PUSH '.SET  VAR=CLASS2        DATA='CLASS2
/*PUSH '.SET  VAR=COMMAND       DATA='COMMAND                          */
/*PUSH '.SET  VAR=COMMAND2      DATA='COMMAND2                         */
  PUSH '.SET  VAR=COMPOPT       DATA='COMPOPT
  PUSH '.SET  VAR=COMSTEP       DATA='COMSTEP
  PUSH '.SET  VAR=DEBUGØ        DATA='DEBUGØ
  PUSH '.SET  VAR=DEBUG1        DATA='DEBUG1
  PUSH '.SET  VAR=DEBUG2        DATA='DEBUG2
  PUSH '.SET  VAR=DEBUG3        DATA='DEBUG3
  PUSH '.SET  VAR=DEBUG4        DATA='DEBUG4
  PUSH '.SET  VAR=DISP          DATA='DISP
  PUSH '.SET  VAR=DSECTL1       DATA='DSECTL1
  PUSH '.SET  VAR=DSECTL2       DATA='DSECTL2
  PUSH '.SET  VAR=DSECTL3       DATA='DSECTL3
  PUSH '.SET  VAR=EXEC          DATA='EXEC
  PUSH '.SET  VAR=INCTYPE       DATA='INCTYPE
  PUSH '.SET  VAR=JDEBUG        DATA='JDEBUG
  PUSH '.SET  VAR=JCTYP         DATA='JCTYP
  PUSH '.SET  VAR=JMODE         DATA='JMODE
  PUSH '.SET  VAR=JNAME         DATA='JNAME
  PUSH '.SET  VAR=JTYPE         DATA='JTYPE
  PUSH '.SET  VAR=JPANLIB       DATA='JPANLIB
  PUSH '.SET  VAR=JPHASE        DATA='JPHASE
  PUSH '.SET  VAR=JS            DATA='JS
  PUSH '.SET  VAR=LCLASS        DATA='LCLASS
  PUSH '.SET  VAR=LCLASS1       DATA='LCLASS1
  PUSH '.SET  VAR=LCLASS2       DATA='LCLASS2
  PUSH '.SET  VAR=LCLASS3       DATA='LCLASS3
  PUSH '.SET  VAR=LCLASS4       DATA='LCLASS4
  PUSH '.SET  VAR=LDEST         DATA='LDEST
  PUSH '.SET  VAR=LDEST1        DATA='LDEST1
  PUSH '.SET  VAR=LDEST2        DATA='LDEST2
  PUSH '.SET  VAR=LDEST3        DATA='LDEST3
  PUSH '.SET  VAR=LDEST4        DATA='LDEST4
  PUSH '.SET  VAR=LDISP         DATA='LDISP
  PUSH '.SET  VAR=LDISP1        DATA='LDISP1
  PUSH '.SET  VAR=LDISP2        DATA='LDISP2
  PUSH '.SET  VAR=LDISP3        DATA='LDISP3
  PUSH '.SET  VAR=LDISP4        DATA='LDISP4
  PUSH '.SET  VAR=LIBDEF1       DATA='LIBDEF1
  PUSH '.SET  VAR=LIBDEF2       DATA='LIBDEF2
  PUSH '.SET  VAR=LINK          DATA='LINK
  PUSH '.SET  VAR=LSTCICS       DATA='LSTCICS
  PUSH '.SET  VAR=PVTYPE        DATA='PVTYPE
  PUSH '.SET  VAR=SQLPARM       DATA='SQLPARM
  PUSH '.SET  VAR=SQLPWD        DATA='SQLPWD
  PUSH '.SET  VAR=SQLUSER       DATA='SQLUSER
```

```
    PUSH '.SET  VAR=STEPD         DATA='STEPD
    PUSH '.SET  VAR=STEPD2        DATA='STEPD2
    PUSH '.SET  VAR=SUBLIB        DATA='SUBLIB
    PUSH '.SET  VAR=TJNAME        DATA='TJNAME
  /*PUSH '.SET  VAR=TYPECOM       DATA='TYPECOM                      */
    PUSH '.SET  VAR=USERID        DATA='USERID

/*********************************************************************/
/***  CAUTION - THE FOLLOWING LINES MUST BE IN THE STACK ORDER      ***/
/***  SHOWN DUE TO LIFO WHEN THEY ARE READ BACK IN.                 ***/
/*********************************************************************/

    PUSH '.* ' JCTYP
    PUSH '.* ' JDEBUG
    PUSH '.* ' JLISTD
    PUSH '.* ' JLISTW
    PUSH '.* ' JVSE
    PUSH '.* ' JPANLIB
    PUSH '.* ' JPHASE
    PUSH '.* ' JPTYP
    PUSH '.* ' JMODE
    PUSH '.* ' JTYPE
    PUSH '.* ' JNAME
    PUSH '.*         D O   N O T   C H A N G E   T H I S   F I L E '

   'EXECIO 'QUEUED() 'DISKW' JNAME' AUXFILE A1 (FINIS'

/*********************************************************************/
/***  SUBMIT THE JCL USING THE AUXILIARY FILE THAT WAS CREATED      ***/
/***  ABOVE.                                                        ***/
/*********************************************************************/

  'VSICP PGM JCL * (DISK' JNAME 'AUXFILE *'

   IF RC ¬= 'Ø' THEN DO
      MSG1 = 'PGMØØ14  Error occurred during submit.'
      MSG2 = '         Contact Help Desk if unable to resolve.'
      CALL DISPØØ  ;  SIGNAL CKRESP  ;  END

   MSG1 = 'PGMØØ15  Submit Completed Successfully.'
   CALL DISPØØ  ;  SIGNAL CKRESP

/*********************************************************************/
/***  DISPLAY MAIN MENU...                                          ***/
/*********************************************************************/
   DISPØØ:

   IF SUBSTR(JNAME,1,2) = 'PF' THEN JNAME = ' '

   CRESP = ' ' ; CPFKEY = ' '
```

```
   ADDRESS ISPEXEC 'DISPLAY PANEL(PGMPØØ)'
   MSG1 = ' '  ;  MSG2 = ' '  ;  SMSG = ' '

   RETURN

/*********************************************************************/
/***  SET INITIAL DEFAULTS IF NO AUXFILE IS FOUND...               ***/
/*********************************************************************/
   DEFAULT:

   MSG1 = 'PGMØØØ2  Auxfile not specified or not found.'
   MSG2 = 'Default settings applied.'
   IF JNAME = '' THEN JNAME = ' '
   IF JTYPE = '' THEN JTYPE = 'COBOL'
   IF JMODE = '' THEN JMODE = 'A '
   JPTYP   = 'CCMD'
   JPHASE  = JNAME
   JPANLIB = '1'                       /* PANVALET LIBRARY           */
   JVSE    = '1'                       /* VSE SUBLIB                 */
   JLISTW  = '4'                       /* LISTINGS WANTED            */
   JLISTD  = '1'                       /* LISTING DESTINATION        */
   JDEBUG  = '1'                       /* DEBUGGER OPTION            */
   JCTYP   = '2'                       /* TYPE OF COBOL              */

   RETURN

/*********************************************************************/
/***  CONVERT OLD-STYLE AUXFILES TO NEW FORMAT...      Ø8/25/97     ***/
/*********************************************************************/
   CONVERT:

 IF   JVSE = '1' | JVSE = '2' THEN RETURN  /* VALID SUBLIB MEANS      */
 ELSE JVSE = '1'                           /* AUXFILE ALREADY CHGD    */

 IF      JLISTW = 'PSC' THEN JLISTW = '1'  /* LISTING-DESIRED         */
 ELSE IF JLISTW = 'SC'  THEN JLISTW = '2'
 ELSE IF JLISTW = 'C'   THEN JLISTW = '3'
 ELSE                        JLISTW = '4'

 IF JLISTD = 'C' THEN JLISTD = '1'         /* LISTING DESTINATION     */
 ELSE                 JLISTD = '2'

 IF JDEBUG = 'I' THEN JDEBUG = '2'         /* DEBUGGER OPTION         */
 ELSE                 JDEBUG = '1'

 IF JCTYPXX = 'YES' THEN JCTYP = '3'       /* COBOL TYPE              */
 ELSE                    JCTYP = 'Ø'       /* SET INVALID TO FORCE    */
                                           /* USER TO CHOOSE          */
   RETURN
```

```
/*********************************************************************/
/***  SET PARMS FOR INTERTEST COMPILE...                           ***/
/*********************************************************************/
   INTRTEST:

 LDISP4  = 'D'                        /* HOLD IN QUEUE FOR DBUGGR     */
 LCLASS4 = 'G'                        /* CLASS DBUGGR WILL LOOK IN    */
 LDEST4  = 'CSQ'                      /* DEST FOR PRE-DBUGGR LSTNG    */

     /*****************************************************************/
     /**  CAUTION  - DEBUGX LINES EXTEND PAST VIEWABLE AREA BELOW    **/
     /*****************************************************************/

   IF INCTYPE = 'PGMACMD' | INCTYPE = 'PGMCCMD' THEN DO
      DEBUG1 = "'$$ØØ LST FDEF=FlØ217,PDEF=PlØ177,RBS=Ø,             X'"
      DEBUG2 = "'$$ØØ DISP="LDISP",CLASS="LCLASS",DEST=(,"LDEST")'"
      END
   ELSE DO
      DEBUG1 = "'$$Ø1 LST FDEF=F1Ø217,PDEF=P1Ø177,RBS=Ø,             X'"
      DEBUG2 = "'$$Ø1 DISP="LDISP",CLASS="LCLASS",DEST=(,"LDEST")'"
      END

 IF JCTYP = '1' THEN DO                          /* ASSEMBLER PGMS     */
    DEBUG3 = "'// EXEC IN25SYMA'"
    DEBUG4 =
"'"JPHASE",LISTER=REF,NOPURGE,POWER=("TJNAME","LCLASS4","LCLASS")'"
      END
   ELSE
    IF JCTYP = '2' | JCTYP = '3' THEN DO         /* C2 OR CVSE PGMS    */
         DEBUG3 = "'// EXEC IN25COB2'"
         DEBUG4 =
"'"TJNAME",LISTER=REF,NOPURGE,POWER=("TJNAME","LCLASS4","LCLASS")'"
         END
    ELSE DO                                      /* DOS/COBOL PGMS     */
         DEBUG3 = "'// EXEC IN25SYMC'"
         DEBUG4 =
"'"TJNAME",LISTER=MAP,NOPURGE,POWER=("TJNAME","LCLASS4","LCLASS")'"
         END

   RETURN

/*********************************************************************/
/***  SET PARMS FOR SIMON COMPILE...                               ***/
/*********************************************************************/
   SIMON:

 LDISP4  = 'D'                        /* HOLD IN QUEUE FOR DBUGGR     */
 LCLASS4 = 'G'                        /* CLASS DBUGGR WILL LOOK IN    */
 LDEST4  = 'CSQ'                      /* DEST FOR PRE-DBUGGR LSTNG    */
```

```
     /*****************************************************************/
     /**  CAUTION  - DEBUGX LINES EXTEND PAST VIEWABLE AREA BELOW    **/
     /*****************************************************************/

 IF INCTYPE = 'PGMABAT' | INCTYPE = 'PGMACALL' | ,
    INCTYPE = 'PGMCBAT' | INCTYPE = 'PGMCCALL' THEN DO
       DEBUG1 = "'$ $$ LST FDEF=F10217,PDEF=P10177,RBS=0,            X'"
       DEBUG2 = "'$ $$ DISP="LDISP",CLASS="LCLASS",DEST=(,"LDEST")'"
         END
 ELSE DO
       DEBUG1 = "'$$00 LST FDEF=F10217,PDEF=P10177,RBS=0,            X'"
       DEBUG2 = "'$$00 DISP="LDISP",CLASS="LCLASS",DEST=(,"LDEST")'"
         END

   DEBUG3 = "'// EXEC SIMONBL,SIZE=SIMONBL'"

IF JCTYP = '1' THEN                              /* ASSEMBLER PGMS     */
DEBUG4 = "'ASM,PJOB="TJNAME",PCLASS="LCLASS4",NOPMAP,PDEST="LDEST4"'"
 ELSE
    IF JCTYP = '2' | JCTYP = '3' THEN            /* C2 OR CVSE PGMS    */
         DEBUG4 =
"'COB2R3,PJOB="TJNAME",PCLASS="LCLASS4",NOPMAP,PDEST="LDEST4"'"
      ELSE                                       /* DOS/COBOL PGMS     */
         DEBUG4 =
"'COB,PJOB="TJNAME",PCLASS="LCLASS4",NOPMAP,PDEST="LDEST4"'"

   RETURN

/*********************************************************************
***  PGM PANEL source code...                                      ***
*********************************************************************/

  )ATTR
    $ TYPE(TEXT) INTENS(LOW)  COLOR(BLUE)  CAPS(ON) SKIP(ON)
    ? TYPE(TEXT) INTENS(HIGH)  COLOR(WHITE) CAPS(OFF)
    # TYPE(OUTPUT) INTENS(LOW) COLOR(YELLOW) CAPS(OFF)
    @ TYPE(INPUT) INTENS(NON)
    ¬ TYPE(TEXT) INTENS(LOW)  COLOR(YELLOW)  CAPS(OFF)
    * TYPE(TEXT) INTENS(LOW)  COLOR(YELLOW)  CAPS(OFF) HILITE(USCORE)
    _ TYPE(INPUT) COLOR(WHITE) CAPS(ON)
  )BODY SMSG(SMSG)
  $PGMP00               ?   Program Generation Menu     $
$&WDATE
  +
  +File Name ==>_JNAME   $  #MSG1                                      $
  +     Type ==>_JTYPE   $  #MSG2                                      $
  +     Mode ==>_Z $       #SMSG                                       $
  +
  +Program Type ==>_JPTYP$
  + ¬CCMD +Cob CICS          ¬ACMD +Asm CICS     ¬DSECT+map(DSECT)
```

```
  + ¬CCMDC+Cob CICS called   ¬ABAT +Asm batch    ¬MAP  +map(physical)
  + ¬CBAT +Cob batch         ¬ACALL+Asm batch called  ¬DYLAK+Dylakor
freeze
  + ¬CCALL+Cob batch called  ¬ASQL +Asm CICS SQL      ¬FDEF +Formdef
  + ¬CSQL +Cob CICS SQL      ¬ASQLB+Asm batch SQL     ¬PDEF +Pagedef
  + ¬CSQLC+Cob CICS SQL called                        ¬OVLY +Overlay
  + ¬CSQLB+Cob batch SQL
  + ¬CSQBC+Cob batch SQL called
  +
  +Phase Name     ==>_JPHASE  $¬NOLINK+to skip lnkedt;  Default is File
Name
  +Panvalet Lib   ==>_Z$       ¬1+Production¬2+Programmer¬3+Data Adm.
  +VSE Sublib     ==>_Z$       ¬1+LIB1.TEST ¬2+LIB1.TEST2
  +Listing Wanted ==>_Z$       ¬1+Panvalet  ¬2+SQL prep  ¬3+CICS
prep¬4+Compile
  +Listing Dest   ==>_Z$       ¬1+&USERID ¬  2+VSE Power
  +Debugger Type  ==>_Z$       ¬1+None      ¬2+Intertest ¬3+Simon
  +Cobol Type     ==>_Z$       ¬1+Non-Cobol ¬2+Cobol/VSE ¬3+Cobol II
¬4+DOS/Cobol
  + ¬Enter+Submit Job  ¬PF3+Exit
  )INIT
    .HELP   = PGMHØØ
    .ZVARS = '(JMODE JPANLIB JVSE JLISTW JLISTD JDEBUG JCTYP)'

    IF (&MSG1 > ' ')
       .ALARM = YES

    IF (&MSG2 > ' ')
       .ALARM = YES

  )PROC

    &CPFKEY = .PFKEY
    &CRESP  = .RESP
    VPUT (CPFKEY CRESP) PROFILE

  )END


*********************************************************************
***  PGM.JCL for initial compile step...                          ***
*********************************************************************

  .IM VSET DEST *
  * $$ JOB JNM=&TJNAME-&JS,DISP=D,CLASS=C,USER=&USERID-.&STEPD2
  * $$ LST FDEF=F1Ø217,PDEF=P1Ø177,RBS=Ø,                          X
  * $$ DISP=&LDISP1-,CLASS=&LCLASS1-,DEST=(,&LDEST1-)
  * $$ PUN DISP=I,CLASS=&CLASS,RBS=Ø,USER=&USERID-.&STEPD
  // JOB &TJNAME-&JS  &COMSTEP
  // EXEC PROC=$$VASGN
```

```
  &DSECTL1
  // EXEC PROC=PVASN&JPANLIB,USER=&USERID
  // EXEC PAN#1
  ++USING PANXØ3
  ++ADD &JNAME-TT,&PVTYPE,NOFORMAT
  DUMMY ENTRY FOR &JNAME-TT
  ++UPDATE &JNAME-TT,Ø,ALL
  .IM &JNAME &JTYPE &JMODE
  --INSERT PUNCH
  .INCLUDE FN=&INCTYPE FT=PROC FM=*
  /*
  /&
  * $$ EOJ


*************************************************************************
***  JCL procs for individual compile types...                        ***
*************************************************************************

.*************************************************************************
.***  ABAT - ASSEMBLER BATCH                                           ***
.*************************************************************************
  $ $$ LST FDEF=F1Ø217,PDEF=P1Ø177,RBS=Ø,                               X
  $ $$ DISP=&LDISP4-,CLASS=&LCLASS4-,DEST=(,&LDEST4-),JNM=&TJNAME
  // JOB &TJNAME-C               *** PGM - ASSEMBLY ***
  // EXEC PROC=$$VASGN
  // EXEC JSPUTL
     JSEP T(&TJNAME-C) U('ROUTE TO &USERID-')
  $*
  // LIBDEF *,SEARCH=&SUBLIB-,CATALOG=&SUBLIB-
  // OPTION CATAL,SXREF
    PHASE &JPHASE,*
  // EXEC ASSEMBLY,SIZE=256K
  --WRITE PUNCH,&JNAME-TT
  --INSERT PUNCH
  $*
  &DEBUG1
  &DEBUG2
  &DEBUG3
  &DEBUG4
  &LINK
  $&


.*************************************************************************
.***  ACMD - ASSEMBLER CICS                                            ***
.*************************************************************************
$ $$ LST FDEF=F1Ø217,PDEF=P1Ø177,RBS=Ø,                                 X
  $ $$ DISP=&LDISP3-,CLASS=&LCLASS3-,DEST=(,&LDEST3-)
  $ $$ PUN DISP=I,CLASS=&CLASS2,RBS=Ø,PRI=7,USER=&USERID-.ASSEM
```

```
  // JOB &TJNAME-C                *** PGM - CICS PREP ***
  // EXEC PROC=$$VASGN
  // EXEC $SYSPCH
  $$ØØ LST FDEF=F1Ø217,PDEF=P1Ø177,RBS=Ø,                              X
  $$ØØ DISP=&LDISP4-,CLASS=&LCLASS4-,DEST=(,&LDEST4-),JNM=&TJNAME
  // JOB &TJNAME-C                *** PGM - ASSEMBLY ***
  // EXEC PROC=$$VASGN
  // EXEC JSPUTL
     JSEP T(&TJNAME-C) U('ROUTE TO &USERID-')
  $$ØØ /*
  // LIBDEF *,SEARCH=&SUBLIB-,CATALOG=&SUBLIB-
  // OPTION SYM,ERRS,CATAL,NODECK,SXREF
    PHASE &JPHASE,*
    INCLUDE DFHEAI
  // EXEC ASSEMBLY,SIZE=256K
  $$ØØ EXIT
  // OPTION DECK
  // EXEC DFHEAP1$,SIZE=256K
  --WRITE PUNCH,&JNAME-TT,/*
  --INSERT PUNCH
  // EXEC $SYSPCH
  $$ØØ /*
  &DEBUG1
  &DEBUG2
  &DEBUG3
  &DEBUG4
  &LINK
  $$ØØ /&
  $$ØØ EXIT
  $&


.*************************************************************************
.***  AMAP - ASSEMBLER MAP                                             ***
.*************************************************************************
$ $$ LST FDEF=F1Ø217,PDEF=P1Ø177,RBS=Ø,                                X
  $ $$ DISP=&LDISP-,CLASS=&LCLASS-,DEST=(,&LDEST-)
  // JOB &TJNAME-C                *** PGM - ASSEMBLY ***
  // EXEC PROC=$$VASGN
  // EXEC JSPUTL
     JSEP T(&TJNAME-C) U('ROUTE TO &USERID-')
  $*
  // LIBDEF *,SEARCH=&SUBLIB-,CATALOG=&SUBLIB-
  // OPTION CATAL,NODECK,SYSPARM='MAP'
    PHASE &JPHASE,*
  // EXEC ASSEMBLY,SIZE=256K
  --WRITE PUNCH,&JNAME-TT
  --INSERT PUNCH
  $*
  &LINK
  $&
```

```
.*************************************************************************
.***  ASQL - ASSEMBLER CICS SQL                                        ***
.*************************************************************************
$ $$ LST FDEF=F10217,PDEF=P10177,RBS=0,                                  X
  $ $$ DISP=&LDISP2-,CLASS=&LCLASS2-,DEST=(,&LDEST2-)
  $ $$ PUN DISP=I,CLASS=&CLASS2,RBS=0,PRI=7,USER=&USERID-.CICS
  // JOB &TJNAME-C                *** PGM - SQL PREP ***
  // EXEC PROC=$$VASGN
  // EXEC $SYSPCH
  $$00 LST FDEF=F10217,PDEF=P10177,RBS=0,                                X
  $$00 DISP=&LDISP3-,CLASS=&LCLASS3-,DEST=(,&LDEST3-)
  $$00 PUN DISP=I,CLASS=&CLASS2,RBS=0,PRI=7,USER=&USERID-.ASSEM
  // JOB &TJNAME-C                *** PGM - CICS PREP ***
  // EXEC PROC=$$VASGN
  // EXEC $SYSPCH
  $$01 LST FDEF=F10217,PDEF=P10177,RBS=0,                                X
  $$01 DISP=&LDISP4-,CLASS=&LCLASS4-,DEST=(,&LDEST4-),JNM=&TJNAME
  // JOB &TJNAME-C                *** PGM - ASSEMBLY ***
  // EXEC PROC=$$VASGN
  // EXEC JSPUTL
     JSEP T(&TJNAME-C) U('ROUTE TO &USERID-')
  $$01 /*
  // LIBDEF *,SEARCH=&SUBLIB-,CATALOG=&SUBLIB-
  // OPTION LISTX,SYM,ERRS,CATAL,NODECK
   PHASE &JPHASE,*
   INCLUDE DFHEAI
  // EXEC ASSEMBLY,SIZE=400K
  $$01 EXIT
  // OPTION NOLIST,NODUMP,DECK
  // EXEC DFHEAP1$,SIZE=400K
  $$00 EXIT
  // OPTION NOLIST,NODUMP,DECK
  // EXEC ARIPRPA,SIZE=AUTO,PARM='NOCHECK,KEEP,PU,BLK,                   X
                 PREP=&JPHASE,&SQLPARM,USERID=&SQLUSER/&SQLPWD'
  --WRITE PUNCH,&JNAME-TT,/*
  --INSERT PUNCH
  // EXEC $SYSPCH
  $$00 /*
  // EXEC $SYSPCH
  $$01 /*
  &DEBUG1
  &DEBUG2
  &DEBUG3
  &DEBUG4
  $$01 /*
   INCLUDE ARIRRTED
  &LINK
  $$01 /&
  $$01 EXIT
```

```
  $$Ø0 /&
  $$Ø0 EXIT
  $&


.************************************************************************
.***  CBAT - COBOL BATCH                                               ***
.************************************************************************
$ $$ LST FDEF=F1Ø217,PDEF=P1Ø177,RBS=Ø,                                  X
  $ $$ DISP=&LDISP4-,CLASS=&LCLASS4-,DEST=(,&LDEST4-),JNM=&TJNAME
  // JOB &TJNAME-C               *** PGM - COMPILE ***
  // EXEC PROC=$$VASGN
  // EXEC JSPUTL
     JSEP T(&TJNAME-C) U('ROUTE TO &USERID-')
  $*
  &LIBDEF1
  &LIBDEF2
  &COMPOPT
    PHASE &JPHASE,*
  &EXEC
  &DEBUGØ
  --WRITE PUNCH,&JNAME-TT
  --INSERT PUNCH
  $*
  &DEBUG1
  &DEBUG2
  &DEBUG3
  &DEBUG4
  &LINK
  $&


.************************************************************************
.***  CCMC - COBOL CICS CALLED                                         ***
.************************************************************************
$ $$ LST FDEF=F1Ø217,PDEF=P1Ø177,RBS=Ø,                                  X
  $ $$ DISP=&LDISP3-,CLASS=&LCLASS3-,DEST=(,&LDEST3-)
  $ $$ PUN DISP=I,CLASS=&CLASS2,RBS=Ø,PRI=7,USER=&USERID-.COMPILE
  // JOB &TJNAME-C               *** PGM - CICS PREP ***
  // EXEC PROC=$$VASGN
  // EXEC $SYSPCH
  $$Ø0 LST FDEF=F1Ø217,PDEF=P1Ø177,RBS=Ø,                                X
  $$Ø0 DISP=&LDISP4-,CLASS=&LCLASS4-,DEST=(,&LDEST4-),JNM=&TJNAME
  $$Ø0 PUN DISP=I,CLASS=&CLASS2,RBS=Ø,PRI=7,USER=&USERID-.CATALOG
  // JOB &TJNAME-C               *** PGM - COMPILE ***
  // EXEC PROC=$$VASGN
  // EXEC $SYSPCH
  $$Ø1 LST FDEF=F1Ø217,PDEF=P1Ø177,RBS=Ø,                                X
  $$Ø1 DISP=&LDISP-,CLASS=&LCLASS-,DEST=(,&LDEST-)
  // JOB &TJNAME-C               *** PGM - CATALOG ***
```

```
  // OPTION LOG
  // EXEC PROC=$$VASGN
  // EXEC JSPUTL
     JSEP T(&TJNAME-C) U('ROUTE TO &USERID-')
  $$Ø1 /*
  &DEBUG1
  &DEBUG2
  // EXEC LIBR
  A S=&SUBLIB
  CATALOG &JPHASE-.OBJ REPLACE=YES
  $$Ø1 EXIT
  &LIBDEF1
  &LIBDEF2
  &COMPOPT
  &EXEC
  $$ØØ EXIT
  // OPTION DECK
  // EXEC DFHECP1$,SIZE=4ØØK,PARM='&LSTCICS'
  --WRITE PUNCH,&JNAME-TT,/*
  --INSERT PUNCH
  // EXEC $SYSPCH
  $$ØØ /*
  // EXEC $SYSPCH
  $$Ø1 /+
  $$Ø1 /*
  $$Ø1 /&
  $$Ø1 EXIT
  $$ØØ /&
  $$ØØ EXIT
  $&


.************************************************************************
.***  CCMD - COBOL CICS                                                ***
.************************************************************************
$ $$ LST FDEF=F1Ø217,PDEF=P1Ø177,RBS=Ø,                                  X
  $ $$ DISP=&LDISP3-,CLASS=&LCLASS3-,DEST=(,&LDEST3-)
  $ $$ PUN DISP=I,CLASS=&CLASS2,RBS=Ø,PRI=7,USER=&USERID-.COMPILE
  // JOB &TJNAME-C               *** PGM - CICS PREP ***
  // EXEC PROC=$$VASGN
  // EXEC $SYSPCH
  $$ØØ LST FDEF=F1Ø217,PDEF=P1Ø177,RBS=Ø,                               X
  $$ØØ DISP=&LDISP4-,CLASS=&LCLASS4-,DEST=(,&LDEST4-),JNM=&TJNAME
  // JOB &TJNAME-C               *** PGM - COMPILE ***
  // EXEC PROC=$$VASGN
  // EXEC JSPUTL
     JSEP T(&TJNAME-C) U('ROUTE TO &USERID-')
  $$ØØ /*
  &LIBDEF1
  &LIBDEF2
```

```
  &COMPOPT
    PHASE &JPHASE,*
    INCLUDE DFHECI
  &EXEC
  $$ØØ EXIT
  // OPTION DECK
  // EXEC DFHECP1$,SIZE=4ØØK,PARM='&LSTCICS'
  --WRITE PUNCH,&JNAME-TT,/*
  --INSERT PUNCH
  // EXEC $SYSPCH
  $$ØØ /*
  &DEBUG1
  &DEBUG2
  &DEBUG3
  &DEBUG4
  &LINK
  $$ØØ /&
  $$ØØ EXIT
  $&


.*************************************************************************
.***  CSQL - COBOL CICS SQL                                            ***
.*************************************************************************
$ $$ LST FDEF=F1Ø217,PDEF=P1Ø177,RBS=Ø,                                  X
  $ $$ DISP=&LDISP2-,CLASS=&LCLASS2-,DEST=(,&LDEST2-)
  $ $$ PUN DISP=I,CLASS=&CLASS2,RBS=Ø,PRI=7,USER=&USERID-.CICS
  // JOB &TJNAME-C               *** PGM - SQL PREP ***
  // EXEC PROC=$$VASGN
  // EXEC $SYSPCH
  $$ØØ LST FDEF=F1Ø217,PDEF=P1Ø177,RBS=Ø,                               X
  $$ØØ DISP=&LDISP3-,CLASS=&LCLASS3-,DEST=(,&LDEST3-)
  $$ØØ PUN DISP=I,CLASS=&CLASS2,RBS=Ø,PRI=7,USER=&USERID-.COMPILE
  // JOB &TJNAME-C               *** PGM - CICS PREP ***
  // EXEC PROC=$$VASGN
  // EXEC $SYSPCH
  $$Ø1 LST FDEF=F1Ø217,PDEF=P1Ø177,RBS=Ø,                               X
  $$Ø1 DISP=&LDISP4-,CLASS=&LCLASS4-,DEST=(,&LDEST4-),JNM=&TJNAME
  // JOB &TJNAME-C               *** PGM - COMPILE ***
  // EXEC PROC=$$VASGN
  // EXEC JSPUTL
     JSEP T(&TJNAME-C) U('ROUTE TO &USERID-')
  $$Ø1 /*
  &LIBDEF1
  &LIBDEF2
  &COMPOPT
   PHASE &JPHASE,*
   INCLUDE DFHECI
  &EXEC
  $$Ø1 EXIT
```

```
  // OPTION NOLIST,NODUMP,DECK
  // EXEC DFHECP1$,SIZE=400K,PARM='&LSTCICS'
  $$00 EXIT
  // OPTION NOLIST,NODUMP,DECK
  // EXEC ARIPRPC,SIZE=AUTO,PARM='NOCHECK,KEEP,APOST,ISOL(USER),PU,BLK,X
                &SQLPARM,PREP=&JPHASE,USERID=&SQLUSER/&SQLPWD'
  --WRITE PUNCH,&JNAME-TT,/*
  --INSERT PUNCH
  // EXEC $SYSPCH
  $$00 /*
  // EXEC $SYSPCH
  $$01 /*
  &DEBUG1
  &DEBUG2
  &DEBUG3
  &DEBUG4
   INCLUDE ARIRRTED
   INCLUDE ARIPADR4
  &LINK
  $$01 /&
  $$01 EXIT
  $$00 /&
  $$00 EXIT
  $&


.*************************************************************************
.***  CSQL - COBOL CICS SQL                                            ***
.*************************************************************************
$ $$ LST FDEF=F10217,PDEF=P10177,RBS=0,                                  X
  $ $$ DISP=&LDISP2-,CLASS=&LCLASS2-,DEST=(,&LDEST2-)
  $ $$ PUN DISP=I,CLASS=&CLASS2,RBS=0,PRI=7,USER=&USERID-.CICS
  // JOB &TJNAME-C                *** PGM - SQL PREP ***
  // EXEC PROC=$$VASGN
  // EXEC $SYSPCH
  $$00 LST FDEF=F10217,PDEF=P10177,RBS=0,                                X
  $$00 DISP=&LDISP3-,CLASS=&LCLASS3-,DEST=(,&LDEST3-)
  $$00 PUN DISP=I,CLASS=&CLASS2,RBS=0,PRI=7,USER=&USERID-.COMPILE
  // JOB &TJNAME-C                *** PGM - CICS PREP ***
  // EXEC PROC=$$VASGN
  // EXEC $SYSPCH
  $$01 LST FDEF=F10217,PDEF=P10177,RBS=0,                                X
  $$01 DISP=&LDISP4-,CLASS=&LCLASS4-,DEST=(,&LDEST4-),JNM=&TJNAME
  // JOB &TJNAME-C                *** PGM - COMPILE ***
  // EXEC PROC=$$VASGN
  // EXEC JSPUTL
     JSEP T(&TJNAME-C) U('ROUTE TO &USERID-')
  $$01 /*
  &LIBDEF1
```

```
  &LIBDEF2
  &COMPOPT
   PHASE &JPHASE,*
   INCLUDE DFHECI
  &EXEC
  $$Ø1 EXIT
  // OPTION NOLIST,NODUMP,DECK
  // EXEC DFHECP1$,SIZE=4ØØK,PARM='&LSTCICS'
  $$ØØ EXIT
  // OPTION NOLIST,NODUMP,DECK
  // EXEC ARIPRPC,SIZE=AUTO,PARM='NOCHECK,KEEP,APOST,ISOL(USER),PU,BLK,X
                &SQLPARM,PREP=&JPHASE,USERID=&SQLUSER/&SQLPWD'
  --WRITE PUNCH,&JNAME-TT,/*
  --INSERT PUNCH
  // EXEC $SYSPCH
  $$ØØ /*
  // EXEC $SYSPCH
  $$Ø1 /*
  &DEBUG1
  &DEBUG2
  &DEBUG3
  &DEBUG4
   INCLUDE ARIRRTED
   INCLUDE ARIPADR4
  &LINK
  $$Ø1 /&
  $$Ø1 EXIT
  $$ØØ /&
  $$ØØ EXIT
  $&


.************************************************************************
.***  OVLY - AFP OVERLAY                                              ***
.************************************************************************
  $ $$ LST FDEF=F1Ø217,PDEF=P1Ø177,RBS=Ø,                              X
  $ $$ DISP=&LDISP-,CLASS=&LCLASS-,DEST=(,&LDEST-)
  // JOB &TJNAME-C               *** PGM - AFPGEN   ***
  // EXEC PROC=$$VASGN
  // EXEC JSPUTL
     JSEP T(&TJNAME-C) U('ROUTE TO &USERID-')
  $*
  // LIBDEF *,SEARCH=(LIB4.AFPT,LIB4.AFPP,PRD2.AFP),CATALOG=LIB4.AFPT
  // ASSGN SYSØ1Ø,IGN
  // EXEC DZIOVRLY,SIZE=AUTO
  --WRITE PUNCH,&JNAME-TT
  --INSERT PUNCH
  $*
  $&
```

```
.*************************************************************************
.***  XDEF - AFP FORMDEF OR PAGEDEF                                    ***
.*************************************************************************
$ $$ LST FDEF=F10217,PDEF=P10177,RBS=0,                                  X
  $ $$ DISP=&LDISP-,CLASS=&LCLASS-,DEST=(,&LDEST-)
  // JOB &TJNAME-C               *** PGM - AFPGEN   ***
  // EXEC PROC=$$VASGN
  // EXEC JSPUTL
     JSEP T(&TJNAME-C) U('ROUTE TO &USERID-')
  $*
  // LIBDEF *,SEARCH=(LIB4.AFPT,LIB4.AFPP,PRD2.AFP)
  // EXEC AKQPPFA,SIZE=AUTO,                                             X
                PARM='FORMLIB=LIB4.AFPT,PAGELIB=LIB4.AFPT,SIZE=2500K'
  --WRITE PUNCH,&JNAME-TT
  --INSERT PUNCH
  $*
  $&


.*************************************************************************
.***  ACALL - ASSEMBLER BATCH CALLED                                   ***
.*************************************************************************
$ $$ LST FDEF=F10217,PDEF=P10177,RBS=0,                                  X
  $ $$ DISP=&LDISP4-,CLASS=&LCLASS4-,DEST=(,&LDEST4-),JNM=&TJNAME
  $ $$ PUN DISP=I,CLASS=&CLASS2,RBS=0,PRI=7,USER=&USERID-.CATALOG
  // JOB &TJNAME-C               *** PGM - ASSEMBLE ***
  // EXEC PROC=$$VASGN
  // EXEC $SYSPCH
  $$00 LST FDEF=F10217,PDEF=P10177,RBS=0,                                X
  $$00 DISP=&LDISP4-,CLASS=&LCLASS4-,DEST=(,&LDEST4-),JNM=&TJNAME
  // JOB &TJNAME-C               *** PGM - CATALOG ***
  // OPTION LOG
  // EXEC PROC=$$VASGN
  // EXEC JSPUTL
     JSEP T(&TJNAME-C) U('ROUTE TO &USERID-')
  $$00 /*
  // EXEC LIBR
  A S=&SUBLIB
  CATALOG &JPHASE-.OBJ REPLACE=YES
  $$00 EXIT
  // LIBDEF *,SEARCH=&SUBLIB-,CATALOG=&SUBLIB-
  // OPTION DECK
  // EXEC ASSEMBLY,SIZE=256K
  --WRITE PUNCH,&JNAME-TT,/*
  --INSERT PUNCH
  &DEBUG1
  &DEBUG2
  &DEBUG3
  &DEBUG4
  // EXEC $SYSPCH
```

```
  $$ØØ /*
  $$ØØ /&
  $$ØØ EXIT
  $&


.************************************************************************
.***  ADSCT - ASSEMBLER DSECT                                         ***
.************************************************************************
  &DSECTL2
  &DSECTL3-*
  // EXEC JSPUTL
     JSEP T(&TJNAME-&JS) U('ROUTE TO &USERID-')
  /*
  // EXEC $SYSPCH
  $$ØØ PUN DISP=D,CLASS=R,RBS=Ø,DEST=(,&USERID)
  $$ØØ JOB &TJNAME-&JS              *** PGM - REFORMAT DSECT         ***
  // EXEC PROC=$$VASGN
  // EXEC MIS2ØX
  $$ØØ EXIT
  // OPTION DECK,SYSPARM='DSECT'
  // EXEC ASSEMBLY,SIZE=256K
  .IM &JNAME &JTYPE *
  /*
  // EXEC $SYSPCH
  $$ØØ /*
  $$ØØ /*
  $$ØØ /&
  $$ØØ EXIT


.************************************************************************
.***  ASQLB - ASSEMBLER SQL BATCH                                     ***
.************************************************************************
$ $$ LST FDEF=F1Ø217,PDEF=P1Ø177,RBS=Ø,                                 X
  $ $$ DISP=&LDISP2-,CLASS=&LCLASS2-,DEST=(,&LDEST2-)
  $ $$ PUN DISP=I,CLASS=&CLASS2,RBS=Ø,PRI=7,USER=&USERID-.ASSEM
  // JOB &TJNAME-C                *** PGM - SQL PREP ***
  // EXEC PROC=$$VASGN
  // EXEC $SYSPCH
  $$ØØ LST FDEF=F1Ø217,PDEF=P1Ø177,RBS=Ø,                              X
  $$ØØ DISP=&LDISP4-,CLASS=&LCLASS4-,DEST=(,&LDEST4-),JNM=&TJNAME
  // JOB &TJNAME-C                *** PGM - ASSEMBLY ***
  // EXEC PROC=$$VASGN
  // EXEC JSPUTL
     JSEP T(&TJNAME-C) U('ROUTE TO &USERID-')
  $$ØØ /*
  // LIBDEF *,SEARCH=&SUBLIB-,CATALOG=&SUBLIB-
  // OPTION CATAL,NODECK
   PHASE &JPHASE,*
```

```
  // EXEC ASSEMBLY,SIZE=400K
  $$00 EXIT
  // EXEC ARIPRPA,SIZE=AUTO,PARM='NOCHECK,KEEP,PU,BLK,                  X
                PREP=&JPHASE,&SQLPARM,USERID=&SQLUSER/&SQLPWD'
  --WRITE PUNCH,&JNAME-TT,/*
  --INSERT PUNCH
  $*
  // EXEC $SYSPCH
  $$00 /*
   INCLUDE ARIPRDID
  &DEBUG1
  &DEBUG2
  &DEBUG3
  &DEBUG4
  &LINK
  $$00 /&
  $$00 EXIT
  $&


.*************************************************************************
  .***  CCALL - COBOL BATCH CALLED                                     ***
.*************************************************************************
$ $$ LST FDEF=F10217,PDEF=P10177,RBS=0,                                  X
  $ $$ DISP=&LDISP4-,CLASS=&LCLASS4-,DEST=(,&LDEST4-),JNM=&TJNAME
  $ $$ PUN DISP=I,CLASS=&CLASS2,RBS=0,PRI=7,USER=&USERID-.CATALOG
  // JOB &TJNAME-C                *** PGM - COMPILE ***
  // EXEC PROC=$$VASGN
  // EXEC $SYSPCH
  $$00 LST FDEF=F10217,PDEF=P10177,RBS=0,                                X
  $$00 DISP=&LDISP-,CLASS=&LCLASS-,DEST=(,&LDEST-)
  // JOB &TJNAME-C                *** PGM - CATALOG ***
  // OPTION LOG
  // EXEC PROC=$$VASGN
  // EXEC JSPUTL
     JSEP T(&TJNAME-C) U('ROUTE TO &USERID-')
  $$00 /*
  // EXEC LIBR
  A S=&SUBLIB
  CATALOG &JPHASE-.OBJ REPLACE=YES
  $$00 EXIT
  &LIBDEF1
  &LIBDEF2
  &COMPOPT
  &EXEC
  &DEBUG0
  --WRITE PUNCH,&JNAME-TT,/*
  --INSERT PUNCH
  &DEBUG1
```

```
  &DEBUG2
  &DEBUG3
  &DEBUG4
  // EXEC $SYSPCH
  $$ØØ /*
  $$ØØ /&
  $$ØØ EXIT
  $&


.************************************************************************
.***  CSQBC - COBOL SQL BATCH CALLED                                  ***
.************************************************************************
$ $$ LST FDEF=F1Ø217,PDEF=P1Ø177,RBS=Ø,                                 X
  $ $$ DISP=&LDISP2-,CLASS=&LCLASS2-,DEST=(,&LDEST2-)
  $ $$ PUN DISP=I,CLASS=&CLASS2,RBS=Ø,PRI=7,USER=&USERID-.COMPILE
  // JOB &TJNAME-C                *** PGM - SQL PREP ***
  // EXEC PROC=$$VASGN
  // EXEC $SYSPCH
  $$ØØ LST FDEF=F1Ø217,PDEF=P1Ø177,RBS=Ø,                              X
  $$ØØ DISP=&LDISP4-,CLASS=&LCLASS4-,DEST=(,&LDEST4-),JNM=&TJNAME
  $$ØØ PUN DISP=I,CLASS=&CLASS2,RBS=Ø,PRI=7,USER=&USERID-.CATALOG
  // JOB &TJNAME-C                *** PGM - COMPILE ***
  // EXEC PROC=$$VASGN
  // EXEC $SYSPCH
  $$Ø1 LST FDEF=F1Ø217,PDEF=P1Ø177,RBS=Ø,                              X
  $$Ø1 DISP=&LDISP-,CLASS=&LCLASS-,DEST=(,&LDEST-),JNM=&TJNAME
  // JOB &TJNAME-C                *** PGM - CATALOG ***
  // OPTION LOG
  // EXEC PROC=$$VASGN
  // EXEC JSPUTL
     JSEP T(&TJNAME-C) U('ROUTE TO &USERID-')
  $$Ø1 /*
  // EXEC LIBR
  A S=&SUBLIB
  CATALOG &JPHASE-.OBJ REPLACE=YES
  $$Ø1 EXIT
  &LIBDEF1
  &LIBDEF2
  &COMPOPT
  &EXEC
  &DEBUGØ
  $$ØØ EXIT
  // EXEC ARIPRPC,SIZE=AUTO,PARM='NOCHECK,KEEP,APOST,ISOL(USER),PU,BLK,X
                 &SQLPARM,PREP=&JPHASE,USERID=&SQLUSER/&SQLPWD'
  --WRITE PUNCH,&JNAME-TT,/*
  --INSERT PUNCH
  &DEBUG1
  &DEBUG2
```

```
  &DEBUG3
  &DEBUG4
  // EXEC $SYSPCH
  $$Ø0 /*
  // EXEC $SYSPCH
  $$Ø1 /*
  $$Ø1 /&
  $$Ø1 EXIT
  $$Ø0 /&
  $$Ø0 EXIT
  $&


.*************************************************************************
.***  CSQLC - COBOL CICS SQL CALLED                                    ***
.*************************************************************************
$ $$ LST FDEF=F10217,PDEF=P1Ø177,RBS=Ø,                                  X
  $ $$ DISP=&LDISP2-,CLASS=&LCLASS2-,DEST=(,&LDEST2-)
  $ $$ PUN DISP=I,CLASS=&CLASS2,RBS=Ø,PRI=7,USER=&USERID-.CICS
  // JOB &TJNAME-C               *** PGM - SQL PREP ***
  // EXEC PROC=$$VASGN
  // EXEC $SYSPCH
  $$Ø0 LST FDEF=F10217,PDEF=P1Ø177,RBS=Ø,                                X
  $$Ø0 DISP=&LDISP3-,CLASS=&LCLASS3-,DEST=(,&LDEST3-)
  $$Ø0 PUN DISP=I,CLASS=&CLASS2,RBS=Ø,PRI=7,USER=&USERID-.COMPILE
  // JOB &TJNAME-C               *** PGM - CICS PREP ***
  // EXEC PROC=$$VASGN
  // EXEC $SYSPCH
  $$Ø1 LST FDEF=F10217,PDEF=P1Ø177,RBS=Ø,                                X
  $$Ø1 DISP=&LDISP4-,CLASS=&LCLASS4-,DEST=(,&LDEST4-),JNM=&TJNAME
  $$Ø1 PUN DISP=I,CLASS=&CLASS2,RBS=Ø,PRI=7,USER=&USERID-.CATALOG
  // JOB &TJNAME-C               *** PGM - COMPILE ***
  // EXEC PROC=$$VASGN
  // EXEC JSPUTL
     JSEP T(&TJNAME-C) U('ROUTE TO &USERID-')
  $$Ø1 /*
  // EXEC $SYSPCH
  $$Ø2 LST FDEF=F10217,PDEF=P1Ø177,RBS=Ø,                                X
  $$Ø2 DISP=&LDISP-,CLASS=&LCLASS-,DEST=(,&LDEST-)
  // JOB &TJNAME-C               *** PGM - CATALOG ***
  // OPTION LOG
  // EXEC PROC=$$VASGN
  // EXEC JSPUTL
     JSEP T(&TJNAME-C) U('ROUTE TO &USERID-')
  $$Ø2 /*
  // EXEC LIBR
  A S=&SUBLIB
  CATALOG &JPHASE-.OBJ REPLACE=YES
  $$Ø2 EXIT
```

```
  &LIBDEF1
  &LIBDEF2
  &COMPOPT
  &EXEC
  $$Ø1 EXIT
  // OPTION NOLIST,NODUMP,DECK
  // EXEC DFHECP1$,SIZE=4ØØK,PARM='&LSTCICS'
  $$ØØ EXIT
  // OPTION NOLIST,NODUMP,DECK
  // EXEC ARIPRPC,SIZE=AUTO,PARM='NOCHECK,KEEP,APOST,ISOL(USER),PU,BLK,X
                &SQLPARM,PREP=&JPHASE,USERID=&SQLUSER/&SQLPWD'
  --WRITE PUNCH,&JNAME-TT,/*
  --INSERT PUNCH
  // EXEC $SYSPCH
  $$ØØ /*
  // EXEC $SYSPCH
  $$Ø1 /*
  &DEBUG1
  &DEBUG2
  &DEBUG3
  &DEBUG4
  $$Ø1 /*
  // EXEC $SYSPCH
  $$Ø2 /+
  $$Ø2 /*
  $$Ø2 /&
  $$Ø2 EXIT
  $$Ø1 /&
  $$Ø1 EXIT
  $$ØØ /&
  $$ØØ EXIT
  $&


.*************************************************************************
.***  DYLAK - DYLAKOR FREEZE & LINK                                    ***
.*************************************************************************
$ $$ LST FDEF=F1Ø217,PDEF=P1Ø177,RBS=Ø,                                  X
  $ $$ DISP=&LDISP-,CLASS=&LCLASS-,DEST=(,&LDEST-)
  // JOB &TJNAME-C                *** PGM - DYLAKOR ***
  .* // EXEC PROC=$$VASGN
  // EXEC JSPUTL
     JSEP T(&TJNAME-C) U('ROUTE TO &USERID-')
  $*
  // LIBDEF *,SEARCH=&SUBLIB-,CATALOG=&SUBLIB-
  // EXEC PROC=$$DYLFRZ
    OPTION FREEZE &JPHASE
  --WRITE PUNCH,&JNAME-TT
  --INSERT PUNCH
```

```
  $*
  // EXEC PROC=$$DYLLNK
  $*
  $&


.************************************************************************
.***  CSQLB - COBOL SQL BATCH                                         ***
.************************************************************************
$ $$ LST FDEF=F10217,PDEF=P10177,RBS=0,                                 X
  $ $$ DISP=&LDISP2-,CLASS=&LCLASS2-,DEST=(,&LDEST2-)
  $ $$ PUN DISP=I,CLASS=&CLASS2,RBS=0,PRI=7,USER=&USERID-.COMPILE
  // JOB &TJNAME-C                *** PGM - SQL PREP ***
  // EXEC PROC=$$VASGN
  // EXEC $SYSPCH
  $$00 LST FDEF=F10217,PDEF=P10177,RBS=0,                               X
  $$00 DISP=&LDISP4-,CLASS=&LCLASS4-,DEST=(,&LDEST4-),JNM=&TJNAME
  // JOB &TJNAME-C                *** PGM - COMPILE ***
  // EXEC PROC=$$VASGN
  // EXEC JSPUTL
     JSEP T(&TJNAME-C) U('ROUTE TO &USERID-')
  $$00 /*
  &LIBDEF1
  &LIBDEF2
  &COMPOPT
   PHASE &JPHASE,*
   INCLUDE ARIPRDID
   INCLUDE ARIPADR4
  &EXEC
  &DEBUG0
  $$00 EXIT
  // EXEC ARIPRPC,SIZE=AUTO,PARM='NOCHECK,KEEP,APOST,ISOL(USER),PU,BLK,X
                &SQLPARM,PREP=&JPHASE,USERID=&SQLUSER/&SQLPWD'
  --WRITE PUNCH,&JNAME-TT,/*
  --INSERT PUNCH
  $*
  // EXEC $SYSPCH
  $$00 /*
  &DEBUG1
  &DEBUG2
  &DEBUG3
  &DEBUG4
  &LINK
  $$00 /&
  $$00 EXIT
  $&
```

Steve Bernard
(USA)

Determining reader/console mode of execution

The subroutine presented here determines whether the calling program was executed from the card reader or the system console. It enables programs written in high-level languages to determine whether they were executed from a card reader, and thus process cards, or, if they were executed from the console, to use accept statements (as in COBOL programs) to process console entries.

One parameter must be passed, consisting of a one-character field which will contain one of the following values on return to the calling program:

- 'R' – Input is from SYSIPT (ie the calling program was executed from the card reader).
- 'C' – Input is from SYSLOG (ie the calling program was executed from the system console).
- 'I' – Input is from SYSLOG (ie the calling program was executed from an ICCF pseudo partition).

CALLING SEQUENCES

The calling sequences are given below.

COBOL

```
CALL 'DPMODE' USING RTNCDE.
```

ALC

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

RPGII

```
CALL 'DPMODE'
                    PARM          RTNCDE
```

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

DPMODE

```
DPMO     TITLE 'DPMODE - 1.Ø - DETERMINES READER/CONSOLE MODE OF
          EXECUTION SUBROUTINE'
DPMODE   CSECT
DPMODE   AMODE 31
DPMODE   RMODE ANY
         SAVE  (14,12)
         BALR  3,Ø
         USING *,3
         ST    13,SAVEAREA+4
         LA    13,SAVEAREA
         B     DPMODEB
*
         DC    C'DPMODE STARTS HERE. ' INSERT EYE CATCHER.
*
DPMODEB  EQU   *
         L     4,Ø(1)              LOAD ADDRESS OF PASSED PARAMETER.
         MVI   RTNCDE,C' '         CLEAR RTNCDE.
         MVC   RTNCDE,Ø(4)         MVE PASSED PARAMETER TO RTNCDE.
         MVI   RTNCDE,C'C'         ASSUME EXECUTION FROM SYSTEM
                                    CONSOLE
         GETFLD FIELD=ICCFPP       GET INTERACTIVE PARTITION FLAG.
         LTR   1,1                 ARE WE RUNNING IN ICCF PSEUDO
                                    PARTIT
         BZ    DPMODES             NO-BRANCH TO DPMODES.
         MVI   RTNCDE,C'I'         INDICATE EXECUTION FROM ICCF
                                    PSEUDO
         B     DPMODER             BRANCH TO DPMODER.
*
DPMODES  EQU   *
         COMRG                     GET PARTITION COMMUNICATIONS
                                    REGION.
         USING COMREG,1            INFORM ASSEMBLER.
         TM    JCSW1,X'2Ø'         WAS INPUT FROM SYSTEM CONSO
                                    (X'38').
         BNO   DPMODER             YES-BRANCH TO DPMODER.
         DROP  1                   (COMREG).
         MVI   RTNCDE,C'R'         INDICATE EXECUTION FROM CARD
                                    READER.
*
```

```
DPMODER  EQU   *
         MVC   Ø(L'RTNCDE,4),RTNCDE
         L     13,SAVEAREA+4
         RETURN (14,12)
*
         DC    C'DPMODE STORAGE HERE. ' INSERT EYE CATCHER.
*
RTNCDE   DC    C' '
SAVEAREA DC    18F'Ø'
*
* PARTITION COMMUNICATIONS REGION. (MAPCOMR).
*
         MAPCOMR ,
*
DPMODEE  DC    X'FF'
*
         END
```

Robert Botsis
Senior Systems Programmer (USA)

REXX/VSE to LE/VSE interface – update

In my article entitled 'REXX/VSE to LE/VSE interface', in the September 1997 issue of *VSE Update*, I mentioned that I had a problem with the CEELOAD macro, described by APAR PQ00422.

The corresponding PTFs UQ04459 for LE/VSE Version 1.1 and UQ04405 for LE/VSE Version 1.4 are now available.

Walter Richters
(Germany)

Detach logical unit

The program presented here was developed under VSE/ESA 2.1.1 as a guest under VM/ESA 2.1, with EPIC as the tape management system.

BACKGROUND

We had encountered a problem during LIBR back-ups where the logical unit of the output tape had to be assigned before LIBR could be executed. Although an EPIC utility is provided to pre-assign the device dynamically and attach the tape drive to the VSE guest virtual machine, the TLBL option to hold the assignment at close is also required in order for LIBR to function properly; this prevents the tape drive from being detached and the tape unloaded. Although VSE/ESA allows CP commands to be issued from the console, the "DETACH" command requires a virtual device address rather than a logical unit number in order for VM/ESA to detach a device.

The two main functions of this program are:

- To determine the device address to which a logical unit number is assigned.
- To then issue a VM/ESA detach command from the VSE/ESA console.

ASSEMBLY NOTES

In order to assemble the source code given below, you must have the PRD1.MACLIB and PRD2.GEN1 sub-libraries in a LIBDEF search chain. You will also need to make changes if you are using any tape device other than 3420 or 3480.

DETACH LOGICAL UNIT

```
TITLE ' DETACH TAPE FROM LOGICAL UNIT ASSGN '
DETLU    CSECT
RØ       EQU   Ø
```

```
R1       EQU   1         SVC AND SYSTEM PARMS
R2       EQU   2         BASE REGISTER
R3       EQU   3         MISC WORK
R4       EQU   4         MISC WORK
R5       EQU   5
R6       EQU   6
R7       EQU   7         DSECT ADDRESSING
R8       EQU   8         BRANCH AND LINK RETURN
R9       EQU   9
R1Ø      EQU   1Ø
R11      EQU   11
R12      EQU   12
R13      EQU   13
R14      EQU   14
R15      EQU   15
         BALR  R2,RØ              ESTABLISH BASE REGISTER
         BCTR  R2,RØ              ADJUST TO FIT ENTRY POINT
         BCTR  R2,RØ              "      "  "   "     "
         USING DETLU,R2
         MVC   REC,REC-1          CLEAR MESSAGE AREA
         TM    Ø(R1),X'8Ø'        NULL PARM STRING ?
         BZ    ERROR1             YES, DISPLAY ERROR
         L     R3,Ø(R1)           NO, GET ADDRESS OF PARM DATA
         SLL   R3,8               CLEAR THE HIGH ORDER BYTE
         SRL   R3,8               "     "   "    "     "
         MVC   PARM,2(R3)         SAVE PARM DATA
         TR    PARM,TRTBL         BLANK OUT THE JUNK
         LA    R4,PARM            POINT TO PARAMETER DATA
         CLC   PARM(1Ø),=C'DETACH=SYS'
         BE    FMT1
         CLC   PARM(1Ø),=C'DETACH SYS'
         BE    FMT1
         CLC   PARM(7),=C'DET=SYS'
         BE    FMT2
         CLC   PARM(7),=C'DET SYS'
         BE    FMT2
         CLC   PARM(3),=C'SYS'
         BE    FMT3
         B     ERROR2             BAD NEWS - NONE OF ABOVE FORMATS
* EXTRACT AND CONVERT SYS NUMBER
FMT1     PACK  D1,PARM+1Ø(3)      GET SYS LOGICAL UNIT
         B     CVB1
FMT2     PACK  D1,PARM+7(3)       FROM ONE OF THESE 3
         B     CVB1
FMT3     PACK  D1,PARM+3(3)       DIFFERENT LOCATIONS.
         B     CVB1
CVB1     CVB   R3,D1              CONVERT PACKED LU TO HEX
         STC   R3,LU+1            FORMAT FOR THE EXTRACT MACRO
```

```
* EXTRACT CUU FROM SYS NUMBER
         EXTRACT ID=PUB,AREA=BUF,LEN=8,SEL=LU
         LA    R7,BUF              ADDRESS AND MAP THE RETURNED INFO
         USING IJBPUB,R7
         CLI   IJBPDEVT,X'ØØ'           NOT ASSGN ?
         BE    ERROR4                   YES, ERROR
         CLI   IJBPDEVT,X'52'           342Ø TAPE DEVICE ?
         BE    OK1                      YES, OK TO CONTINUE
         CLI   IJBPDEVT,X'54'           348Ø TAPE DEVICE ?
         BE    OK1                      YES, OK TO CONTINUE
         B     ERROR3                   NO, THATS A PROBLEM
OK1      EQU   *
         UNPK  CCUU(3),IJBPCHAN(2)      CHANNEL ADDRESS
         UNPK  CCUU+2(3),IJBPDEVN(2)    UNIT ADDRESS
         TR    CCUU(4),HXTBL            MAKES PRINTABLE HEX
* ISSUE CP DETACH COMMAND FROM VSE/ESA CONSOLE
         MVC   IPT,IPT-1                BLANK OUT CONSOLE BUFFER
         MVC   IPT(11),=C'* CP DETACH' VM COMMAND FROM VSE
         MVC   IPT+12(4),CCUU           VIRTUAL DEVICE ADDRESS
         SLR   RØ,RØ                    ZERO REGISTER
         LA    R1,COMMAND               ADDRESS OF COMMAND
         SVC   3Ø                       ISSUE VSE AR COMMAND
         MVC   REC(9),=C'DETACHLU-'     ID MYSELF AND THE
         MVC   REC+9(16),IPT             COMMAND I ISSUED
         BAL   R8,DSPLY                 DISPLAY ON VSE CONSOLE
         B     EOJ                      THATS ALL FOLKS
ERROR1   MVC   REC(41),=C'DETACHLU-EXEC STATEMENT PARM DATA MISSING'
         BAL   R8,DSPLY                 DISPLAY ON VSE CONSOLE
         B     EOJ                      AND EXIT
ERROR2   MVC   REC(43),=C'DETACHLU-PARM DATA ON EXEC STMT. IS
               INVALID'
         BAL   R8,DSPLY                 DISPLAY ON VSE CONSOLE
         B     EOJ                      AND EXIT
ERROR3   MVC   REC(42),=C'DETACHLU-DEVICE NOT TAPE, NO DETACH ISSUED'
         BAL   R8,DSPLY                 DISPLAY ON VSE CONSOLE
         B     EOJ                      AND EXIT
ERROR4   MVC   REC(39),=C'DETACHLU-LU NOT ASSGN, NO DETACH ISSUED'
         BAL   R8,DSPLY                 DISPLAY ON VSE CONSOLE
         B     EOJ                      AND EXIT
DSPLY    LA    R1,CONSOLE          ADDRESS OF CCB
EXCP     SVC   Ø                   START I/O
WAIT     TM    2(R1),X'8Ø'         COMPLETE ?
         BO    Ø(R8)               YES, RETURN
         SVC   7                   NO, WAIT
EOJ      SVC   14                  ITS ALL OVER NOW
         LTORG
D1       DS    D             WORKAREA
LU       DC    X'Ø1ØØ'       LOGICAL UNIT CLASS AND NUMBER FOR
                             EXTRACT
```

```
CCUU     DC    CL4'0000'     CHANNEL AND DEVICE NUMBER
         DS    D             OVERFLOW FROM ABOVE AND ALIGNMENT BELOW
         DC    CL8'CONSOLE '
CONSOLE  CCB   SYSLOG,CCW0
CCW0     CCW   X'09',REC,X'00',60      SYSLOG WRITE
         DS    D
PARM     DS    CL16
COMMAND  DC    AL2(L'IPT)         CONSOLE COMMAND THAT
         DC    AL4(IPT)            WILL BE PASSED TO VSE
         DC    X'40'
IPT      DS    CL60
         DC    CL60' '            PAD BLANKS
*                0 1 2 3 4 5 6 7 8 9 A B C D E F
TRTBL    DC    X'40404040404040404040404040404040'    0
         DC    X'40404040404040404040404040404040'    1
         DC    X'40404040404040404040404040404040'    2
         DC    X'40404040404040404040404040404040'    3
         DC    X'40404040404040404040A4B4C4D4E4F'     4
         DC    X'50404040404040404040405A5B5C5D5E40'  5
         DC    X'60614040404040404040404040404040'    6
         DC    X'40404040404040404040407A7B7C7D7E7F'  7
         DC    X'40404040404040404040404040404040'    8
         DC    X'40404040404040404040404040404040'    9
         DC    X'40404040404040404040404040404040'    A
         DC    X'40404040404040404040404040404040'    B
         DC    X'40C1C2C3C4C5C6C7C8C9404040404040'    C
         DC    X'40D1D2D3D4D5D6D7D8D9404040404040'    D
         DC    X'4040E2E3E4E5E6E7E8E9404040404040'    E
         DC    X'F0F1F2F3F4F5F6F7F8F9404040404040'    F
         DC    X'F0F1F2F3F4F5F6F7F8F9C1C2C3C4C5C6'    HEX 2 EBCDIC
HXTBL    EQU   *-256
BUF      DS    CL8
         DC    X'40'
REC      DS    CL60
         IJBPUB
         END
```

EXAMPLE OF USAGE

The following shows how the program might be used.

```
// JOB SYSBACK VSE/ESA LIBRARY BACKUP IN A VM/ESA ENVIRONMENT
 // TLBL VSELIB,'LIBR.BACKUP',,,,,,72    <--this EPIC option is to
                                             prevent LUB allocation
                                             and hold the assignment
                                             at close time.
 EXEC PGM=TSIDASS                         <--EPIC pre-assgn utility
                                             program
```

```
 OPEN VSELIB,SYSØ5Ø,OUTPUT
/*
EXEC PGM=LIBR
 BACKUP LIB=IJSYSRS  TLABEL=VSELIB  TAPE=SYSØ5Ø
/*
* Detach and unload tape that is currently assigned to SYSØ5Ø
EXEC PGM=DETACHLU,SIZE=4K,PARM='DETACH=SYSØ5Ø'
/&
```

Robert Payne
Senior Systems Programmer (USA)

VSE Update contributions

Tell us what you have done to make working with VSE easier or quicker at your site. We welcome contributions from VSE novices as well as from more experienced users, and are keen to receive anything from very short 'hints and tips' type articles to longer discussion articles and example code.

Articles for *VSE Update* can be sent to the editor, Fiona Hewitt, at any of the addresses shown on page 2. Alternatively, articles can be 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).

March 1991 – December 1997 index

Items below are references to articles that have appeared in *VSE Update* since March 1991. References show the issue number followed by the page number(s). Individual copies of all issues from that date are available.

IBM has announced VSE/ESA 2.3, which provides Web serving and browser access to VSE data, as well as much-needed integration of TCP/IP networking. A CICS Transaction Server for VSE/ESA is also promised for 1998.

In addition, a new package, the Enterprise Server Offering for VSE, includes the option of OS/390 2.4, plus the latest server technology and current release of VSE, along with services.

For further information, contact your local IBM representative.

* * *

Recently announced is CrossAccess for VSE middleware, which accesses data residing in VSAM, DL/I, sequential, and other VSE data sources. Support for 16- and 32-bit ODBD clients enables integration with any of more than 400 popular ODBC-compliant products. Departmental pricing for CrossAccess starts at $30,000.

For further information, contact:
CROSS ACCESS Corporation, One Tower Lane, Suite 2410, Oakbrook Terrace, IL 60181, USA. Telephone: (630) 954 0500.

* * *

IBM has announced Version 7.6 of its Advanced Communications Function/ Network Control Program for VSE, OS/390, and VM. New features include IP Internal Coupling, RIP Version 2 support, additional Frame Relay traffic management enhancements, call connection balancing in duplicate Token-Ring Coupler (TIC) environments, APPN refinements, and more network management mechanisms.

The company has also announced Millennium Language Extensions (MLE) for changing COBOL and PL/I applications to handle Year 2000 dates. The technology, to be integrated into upcoming releases of its COBOL and PL/I compilers, will automate date century windowing.

MLE for PL/I and COBOL for VSE compilers will be available in the first half of 1998. Pricing will be announced when the products are available.

For further information, contact your local IBM representative.

* * *

Compute (Bridgend) has announced Release 9.8 of SELCOPY, its Year 2000 compliant file manipulation utility.

The company has also announced Release 9.8 of CBLVCAT, its VSAM catalog tuning display utility, which, in addition to being Year 2000 compliant, has in-storage catalog processing for improved performance. VRDS and local timestamp reporting are also included.

For further information, contact:
Compute (Bridgend), 8 Merthyr Mawr Road, Bridgend, Wales, CF31 3NH, UK. Telephone: (1656) 652222.
Compute (Bridgend), 38 Guided Court, Rexdale, ON, M9V 4K6, Canada.