2

MVS update

Special edition

In this issue

A xephon PUBLICATION

MVS Update

Published by

Xephon Inc
PO Box 550547
Dallas, Texas 75355
USA

Phone: 214-340-5690
Fax: 214-341-7081

Editor

Trevor Eddolls
E-mail: trevore@xephon.com

Publisher

Colin Smith
E-mail: info@xephon.com

Subscriptions and back-issues

A year's subscription to *MVS Update*, comprising twelve monthly issues, costs $505.00 in the USA and Canada; £340.00 in the UK; £346.00 in Europe; £352.00 in Australasia and Japan; and £350.00 elsewhere. In all cases the price includes postage. Individual issues, starting with the January 2000 issue, are available separately to subscribers for £29.00 ($43.50) each including postage.

MVS Update on-line

Code from *MVS Update*, and complete issues in Acrobat PDF format, can be downloaded from our Web site at http://www.xephon.com/mvs; you will need to supply a word from the printed issue.

Contributions

When Xephon is given copyright, articles published in *MVS Update* are paid for at the rate of $160 (£100 outside North America) per 1000 words and $80 (£50) per 100 lines of code for the first 200 lines of original material. The remaining code is paid for at the rate of $32 (£20) per 100 lines. To find out more about contributing an article, without any obligation, please download a copy of our *Notes for Contributors* from www.xephon.com/nfc.

Printed in England.

ESCON Director display utility

INTRODUCTION

ESCON Director, sometimes referred to as a ‘dynamic switch’, is a switch that acts as a communications hub for ESCON channels.

It provides the capability to physically interconnect any two links that are attached to it. Such a connection between two ports provides simultaneous two-way information transfer. When a connection is established, the two ports and their respective point-to-point links are connected so that frames received by one of the ports are passed transparently to the other port. Such a connection can be either static or dynamic.

ESCON Directors have a major role in MVS data centres. Systems programmers have to manage them in order to plan their hardware configurations.

Because they are located in computer rooms, it is often painful to get their active configuration from their console – you have to move!

Of course, you can use the MVS D M=SWITCH command to get detailed information about a specific ESCON Director port, but there is no standard utility to get a global view.

The utility described in this article is a REXX program that retrieves an ESCON Director configuration from D M=SWITCH commands and displays it on an ISPF panel.

ESCDCONF REXX

```
/* REXX                                                    */
/* REXX routine to display ESCD configuration              */
/*                                                         */
WAIT_TIME = 2                /* wait time for console service */
s = p                        /* default sort key              */
"ISPEXEC TBCREATE ESCDCONF
         NAMES(PORT PORTD STATUS ND)
         REPLACE"
"CONSPROF SOLDISPLAY(NO) UNSOLDISPLAY(NO)",
```

```
         "SOLNUM(9999) UNSOLNUM(Ø)"
"CONSOLE ACTIVATE"
IF RC <> Ø THEN
  DO
   "CONSOLE DEACTIVATE"
   IF RC <> Ø THEN
    DO
     SAY '*** USERID' USERID() 'NEEDS CONSOLE AUTHORITY'
     EXIT 8
    END
  END
"ISPEXEC TBDISPL ESCDCONF PANEL(ESCDØØ) CURSOR(DEVN)"
DISPRC = RC
DO WHILE DISPRC = Ø             /* until PF3 */
 IF WORDS(ZCMD) >= 1 THEN       /* locate ? */
   DO
     CMD     = WORD(ZCMD,1)
     PORTX   = LEFT(WORD(ZCMD,2),2,'Ø')
     IF (PORTX > "FF" AND PORTX < "ØØ") THEN PORTX = "FF"
     PORTD = X2D(PORTX)
     SELECT
      WHEN CMD = L THEN
        DO
          "ISPEXEC TBTOP  ESCDCONF"
          "ISPEXEC TBSCAN ESCDCONF ARGLIST(PORTD) CONDLIST(GE)"
          "ISPEXEC TBDISPL ESCDCONF PANEL(ESCDØØ)"
          disprc = rc
        END
      OTHERWISE
        DO
        END
     END
   END
  ELSE
   DO
     "ISPEXEC TBCREATE ESCDCONF
              NAMES(PORT PORTD STATUS ND)
              REPLACE"
     P1X = SP
     P2X = EP
     ESCD_ADDR = DEVN
     sortkey = s
     /* CHECK ESCD DEVICE NUMBER */
     CMD = "D U,,,"ESCD_ADDR",1"
     CARTVAL = USERID()||TIME()
     "CONSOLE SYSCMD("CMD") CART("CARTVAL")"
     GET_RC = GETMSG('RESP.','SOL',CARTVAL,,WAIT_TIME)
     IF WORD(RESP.3,2) = "SWCH" THEN
       DO
        P1D = X2D(P1X)
```

```
        P2D = X2D(P2X)
        /* CHECK ESCD PORT RANGE */
        IF P1D > P2D THEN
          DO
           "ISPEXEC SETMSG MSG(ESCDØØ2E)"
          END
         ELSE
          DO
           DO ID = P1D TO P2D
            IX = D2X(ID)
            CMD = "D M=SWITCH("ESCD_ADDR","IX")"
            CARTVAL = USERID()||TIME()
            "CONSOLE SYSCMD("CMD") CART("CARTVAL")"
            GET_RC = GETMSG('RESP.','SOL',CARTVAL,,WAIT_TIME)
            PORT = RIGHT(IX,2,'Ø')
            PORTD = ID
            PARSE VAR RESP.2 "STATUS=" STATUS
            PARSE VAR RESP.3 "NODE = " ND
            "ISPEXEC TBADD ESCDCONF"
           END
           "ISPEXEC TBTOP   ESCDCONF"
          END
       END
     ELSE
       DO     /* it is not a ESCD devnum */
        "ISPEXEC SETMSG MSG(ESCDØØ1E)"
       END
    select
     when sortkey = "P" then sk = "PORTD"
     when sortkey = "S" then sk = "STATUS,c,A,portd,n,A"
     when sortkey = "N" then sk = "ND,c,A,portd,n,A"
    end
    "ISPEXEC TBsort  ESCDCONF fields("sk")"
    "ISPEXEC TBDISPL ESCDCONF PANEL(ESCDØØ) CURSOR(DEVN)"
    DISPRC = RC
    "ISPEXEC CONTROL DISPLAY SAVE"
   END
END
"CONSOLE DEACTIVATE"
```

ESCD01 ISPF PANEL

```
)ATTR
   ! TYPE(OUTPUT) INTENS(LOW) JUST(LEFT)
   ¢ TYPE(OUTPUT) INTENS(HIGH) JUST(LEFT)
   # TYPE(TEXT) COLOR(RED) INTENS(HIGH)
   \ TYPE(TEXT) COLOR(YELLOW) INTENS(HIGH)
   % TYPE(TEXT) COLOR(GREEN) INTENS(HIGH)
   $ TYPE(TEXT) SKIP(ON) INTENS(LOW)
```

```
)BODY EXPAND(@@)
+@-@#Escon Director Configuration+@-@+
$\COMMAND%===>_ZCMD                                          %SCROOL
===>_SAMT+
$
$+ESCD devnum%===>_devn+ Start port%===>_sp+ End port%===>_ep+Sort
key%===>_s+
$
$%Port+%DCM Status+                %Node Descriptor
$
)MODEL
$!Z   +¢Z                      +  !Z                                  +
)INIT
 .ZVARS = '(port status nd)'
 &ZTDMARK = '****************************** BOTTOM OF DATA +
******************+
*************'
)PROC
  IF (.RESP = ENTER)
   VER (&devn,NONBLANK)
   VER (&sp,NONBLANK,hex)
   VER (&ep,NONBLANK,hex)
   VER (&s,list,P,S,N)
)help field(devn) msg(escd001h)
      field(sp) msg(escd002h)
      field(ep) msg(escd002h)
      field(s)  msg(escd003h)
)END
```

SAMPLE DISPLAY

In order to use ESCDCONF, you need to have TSO CONSOLE authority.

```
------------------ Escon Director Configuration ------ Row 1 to 16 of 16
 COMMAND ===>                                          SCROLL ===> PAGE

ESCD devnum ===> 9001 Start port ===> D0 End port ===> DF Sort key ===>P

  Port  DCM Status                  Node Descriptor

  D0    NOT ATTACHED                UNKNOWN
  D1    CHANNEL ATTACHED            002064.2C5.IBM.51.000000069150
  D2    CHANNEL ATTACHED            002084.304.IBM.83.0000000292CA
  D3    NOT DCM ELIGIBLE            003490.A20.STK.10.000000008867
  D4    CHANNEL ATTACHED            002064.2C5.IBM.51.000000069150
  D5    NOT ATTACHED                UNKNOWN
  D6    NOT DCM ELIGIBLE            003490.C22.IBM.77.0000000D5350
  D7    NOT ATTACHED                UNKNOWN
```

```
D8    CHANNEL ATTACHED            002064.2C5.IBM.51.000000069150
D9    CHANNEL ATTACHED            002064.2C5.IBM.51.000000069150
DA    NOT DCM ELIGIBLE            003746.900.IBM.57.000000092804
DB    CHANNEL ATTACHED            002064.2C5.IBM.51.000000069150
DC    OFFLINE, PORT ATTACHMENT    UNKNOWN
DD    NOT DCM ELIGIBLE            003490.A10.STK.03.000000310712
DE    CHANNEL ATTACHED            002084.304.IBM.83.0000000292CA
DF    CHANNEL ATTACHED            002064.2C5.IBM.51.000000069150
```

Alexandre Goupil
Systems Programmer (France)

Possible rounding errors in COBOL on the mainframe

INTRODUCTION

On our site the majority, if not all, of our programs are developed in a Windows client/server environment. Most of these programs are used in the client/server world for online transactions. The batch environment is made up of transferred source recompiled on the mainframe.

Recently, by coincidence, it was discovered that, for the exact same time period, there was a discrepancy between the computed value in the client/server world and that of the mainframe. This was at first difficult to understand because the source was the same, it could only be down to differences in interpretation in the different environments.

After much searching, the location of the error was discovered to be rounding errors in the compiled COBOL on the mainframe.

PROBLEM

Various variables in the system are stored in one form and then used in this form or a derivative of this form. For example, a percentage would be stored as 12.5 but then used as 0.125. This method of

storage and usage is also used by us to work to significant decimal places. To enable this, the COMPUTE ROUNDED statement is used. It is used in such a way as to shift variable contents significant positions right or left. This is done by dividing or multiplying by an exponential expression of 10 (the factor being a whole number), eg:

```
COMPUTE A ROUNDED = B / (10 ** factor)
```

where *factor* is a whole number.

When B had a value of 9.9875 this was rounded to 9.988 in the client/server environment and truncated to 9.987 on the mainframe. This rounding error, through future multiplication, produced a difference and a loss for us of about 10 pence on a quarter yearly insurance policy.

Further investigation has determined when the exponential expression *factor* is defined with a decimal point, eg FACTOR PIC 9V9, then this rounding error occurs when the digit after the last significant digit is 5 (0 to 4 truncated as expected, 6 to 9 rounded as expected, but 5 is truncated, instead of rounded).

In this article I have included a test program to allow the readers to determine whether this problem could also occur at their site. The results when displayed should, in an error-free case, all be the same – that is 9.98 (0.00000998) and not as we have experienced with the occasional 9.97 (0.00000997).

CONCLUSION

This problem may be unique to us. I have documented it and have reported it to IBM. The problem occurs when we use either the COBOL for MVS V2R2 compiler or the newer Enterprise COBOL V3R2.

The discrepancy may not appear to be much at first, but it brings with it other problems that need to be addressed:

- If it is left, it will, every now and again, create a discrepancy in the accounting system, which will need to be documented.

  We can get over the problem by making a MOVE to a factor variable defined without a decimal point (we know that it is

always a whole number at our site). The change however will probably have the effect that somewhere else in the system a balancing discrepancy would then occur and have to be accounted for.

- If we simply change the code, the subsequent change in a customer's premiums could result in their being able to cancel their policy because of an unannounced premium increase.
- If it is an IBM problem, then a fix could cause similar problems to our work-around and perhaps can be applied only between years to avoid accounting discrepancies for a specific accounting period.

A test program and results follow (note: the test program requires no input).

TEST PROGRAM SOURCE CODE

```
IDENTIFICATION DIVISION.
*****************************************************************
 PROGRAM-ID. COMPUØØ1.
 *================================================================
 ENVIRONMENT DIVISION.
 *================================================================
 *****************************************************************
 CONFIGURATION SECTION.
 *****************************************************************
 SPECIAL-NAMES.
     DECIMAL-POINT IS COMMA.
*
 *================================================================
 DATA DIVISION.
 *================================================================
*
 *****************************************************************
 WORKING-STORAGE SECTION.
 Ø1 TEST-VARIABLES.
    Ø5 IVAR-1            PIC S9(1Ø)V9(8).
    Ø5 IVAR-2            PIC S9(1Ø)V9(8).
    Ø5 OVAR-1            PIC S9(1Ø)V9(8).
    Ø5 OVAR-2            PIC S9(1Ø)V9(8).
    Ø5 OVAR-3            PIC S9(1Ø)V9(8).
    Ø5 OVAR-4            PIC S9(1Ø)V9(8).
    Ø5 OVAR-5            PIC S9(1Ø)V9(8).
    Ø5 OVAR-6            PIC S9(1Ø)V9(8).
    Ø5 OVAR-7            PIC S9(1Ø)V9(8).
```

```
    Ø5 OVAR-8            PIC S9(1Ø)V9(8).
    Ø5 OVAR-9            PIC S9(1Ø)V9(8).
    Ø5 OVAR-1Ø           PIC S9(1Ø)V9(8).
    Ø5 OVAR-11           PIC S9(1Ø)V9(8).
    Ø5 OVAR-12           PIC S9(1Ø)V9(8).
    Ø5 FACTOR-1          PIC S9(1Ø)V9(8).
    Ø5 FACTOR-2          PIC 9.
    Ø5 FACTOR-3          PIC 9V9.
    Ø5 FACTOR-4          PIC S9(4).
    Ø5 D-IVAR-1          PIC -Z(9)9,9(8).
    Ø5 D-IVAR-2          PIC -Z(9)9,9(8).
    Ø5 D-OVAR-1          PIC -Z(9)9,9(8).
    Ø5 D-OVAR-2          PIC -Z(9)9,9(8).
    Ø5 D-OVAR-3          PIC -Z(9)9,9(8).
    Ø5 D-OVAR-4          PIC -Z(9)9,9(8).
    Ø5 D-OVAR-5          PIC -Z(9)9,9(8).
    Ø5 D-OVAR-6          PIC -Z(9)9,9(8).
    Ø5 D-OVAR-7          PIC -Z(9)9,9(8).
    Ø5 D-OVAR-8          PIC -Z(9)9,9(8).
    Ø5 D-OVAR-9          PIC -Z(9)9,9(8).
    Ø5 D-OVAR-1Ø         PIC -Z(9)9,9(8).
    Ø5 D-OVAR-11         PIC -Z(9)9,9(8).
    Ø5 D-OVAR-12         PIC -Z(9)9,9(8).
    Ø5 D-FACTOR-1        PIC -Z(9)9,9(8).
    Ø5 D-FACTOR-2        PIC 9.
    Ø5 D-FACTOR-3        PIC 9,9.
    Ø5 D-FACTOR-4        PIC -Z(3)9.
*****************************************************************
 LINKAGE SECTION.
*****************************************************************
*===============================================================
 PROCEDURE DIVISION.
*===============================================================
*****************************************************************
 MAIN SECTION.
*****************************************************************
          PERFORM ROUNDING-Ø1
          GOBACK.
     EXIT.
*****************************************************************
*
*****************************************************************
 ROUNDING-Ø1 SECTION.
*****************************************************************
*       ************************************************
*       INITIALISE AND DISPLAY
*       ************************************************
        MOVE 9,975           TO IVAR-1
        MOVE 2               TO IVAR-2
        DISPLAY ' ** INITIAL VALUES ** '
        DISPLAY 'IVAR-1 = ' IVAR-1
```

```
           DISPLAY 'IVAR-2 = ' IVAR-2
      *    ************************************************
      *    COMPUTE ROUNDED
      *    ************************************************
      *    ************************************************
      *    TEST CASE 1
      *    ************************************************
           COMPUTE FACTOR-4 = 8 - IVAR-2
           COMPUTE OVAR-1 ROUNDED =
                   IVAR-1 / (1Ø ** FACTOR-4)
      *    ************************************************
      *    TEST CASE 2
      *    ************************************************
           COMPUTE OVAR-3 ROUNDED =
                   IVAR-1 / (1Ø ** (8 - IVAR-2))
      *    ************************************************
      *    TEST CASE 3
      *    ************************************************
           COMPUTE OVAR-5 ROUNDED =
                   IVAR-1 / (1Ø ** 6)
      *    ************************************************
      *    TEST CASE 4
      *    ************************************************
           COMPUTE FACTOR-1 = 8 - IVAR-2
           COMPUTE OVAR-7 ROUNDED =
                   IVAR-1 / (1Ø ** FACTOR-1)
      *    ************************************************
      *    TEST CASE 5
      *    ************************************************
           COMPUTE FACTOR-2 = 8 - IVAR-2
           COMPUTE OVAR-9 ROUNDED =
                   IVAR-1 / (1Ø ** FACTOR-2)
      *    ************************************************
      *    TEST CASE 6
      *    ************************************************
           COMPUTE FACTOR-3 = 8 - IVAR-2
           COMPUTE OVAR-11 ROUNDED =
                   IVAR-1 / (1Ø ** FACTOR-3)
      *    ************************************************
      *    COMPUTE BACK
      *    ************************************************
           COMPUTE OVAR-2 =
                   OVAR-1 * (1Ø ** FACTOR-4)
           COMPUTE OVAR-4 =
                   OVAR-3 * (1Ø ** (8 - IVAR-2))
           COMPUTE OVAR-6 =
                   OVAR-5 * (1Ø ** 6)
           COMPUTE OVAR-8 =
                   OVAR-7 * (1Ø ** FACTOR-1)
           COMPUTE OVAR-1Ø =
                   OVAR-9 * (1Ø ** FACTOR-2)
```

```
        COMPUTE OVAR-12 =
                OVAR-11 * (1Ø ** FACTOR-3)
*       ************************************************
*       DISPLAY AFTER
*       ************************************************
        MOVE IVAR-1 TO D-IVAR-1
        MOVE IVAR-2 TO D-IVAR-2
        MOVE OVAR-1 TO D-OVAR-1
        MOVE OVAR-2 TO D-OVAR-2
        MOVE OVAR-3 TO D-OVAR-3
        MOVE OVAR-4 TO D-OVAR-4
        MOVE OVAR-5 TO D-OVAR-5
        MOVE OVAR-6 TO D-OVAR-6
        MOVE OVAR-7 TO D-OVAR-7
        MOVE OVAR-8 TO D-OVAR-8
        MOVE OVAR-9 TO D-OVAR-9
        MOVE OVAR-1Ø TO D-OVAR-1Ø
        MOVE OVAR-11 TO D-OVAR-11
        MOVE OVAR-12 TO D-OVAR-12
        MOVE FACTOR-1 TO D-FACTOR-1
        MOVE FACTOR-2 TO D-FACTOR-2
        MOVE FACTOR-3 TO D-FACTOR-3
        MOVE FACTOR-4 TO D-FACTOR-4
        DISPLAY ' **** RETAINED INPUT VALUES ****** '
        DISPLAY 'IVAR-1 = ' D-IVAR-1
        DISPLAY 'IVAR-2 = ' D-IVAR-2
        DISPLAY ' ******* RESULTS ******** '
        DISPLAY 'TEST CASE 1'
        DISPLAY '----------'
        DISPLAY 'OVAR-1= ' D-OVAR-1
        DISPLAY 'OVAR-2= ' D-OVAR-2
        DISPLAY 'TEST CASE 2'
        DISPLAY '----------'
        DISPLAY 'OVAR-3 = ' D-OVAR-3
        DISPLAY 'OVAR-4 = ' D-OVAR-4
        DISPLAY 'TEST CASE 3'
        DISPLAY '----------'
        DISPLAY 'OVAR-5 = ' D-OVAR-5
        DISPLAY 'OVAR-6 = ' D-OVAR-6
        DISPLAY 'TEST CASE 4'
        DISPLAY '----------'
        DISPLAY 'OVAR-7 = ' D-OVAR-7
        DISPLAY 'OVAR-8 = ' D-OVAR-8
        DISPLAY 'TEST CASE 5'
        DISPLAY '----------'
        DISPLAY 'OVAR-9 = ' D-OVAR-9
        DISPLAY 'OVAR-1Ø= ' D-OVAR-1Ø
        DISPLAY 'TEST CASE 6'
        DISPLAY '----------'
        DISPLAY 'OVAR-11= ' D-OVAR-11
        DISPLAY 'OVAR-12= ' D-OVAR-12
```

```
            DISPLAY '----------'
            DISPLAY 'FACTOR-1 = ' D-FACTOR-1
            DISPLAY 'FACTOR-2 = ' D-FACTOR-2
            DISPLAY 'FACTOR-3 = ' D-FACTOR-3
            DISPLAY 'FACTOR-4 = ' D-FACTOR-4
         EXIT.
      *
      *
```

OUTPUT FROM TEST PROGRM

```
** INITIAL VALUES **
IVAR-1 = 00000000099750000ä
IVAR-2 = 00000000020000000ä
 **** RETAINED INPUT VALUES ******
IVAR-1 =           9,97500000
IVAR-2 =           2,00000000
 ******* RESULTS ********
TEST CASE 1
----------
OVAR-1=           0,00000998
OVAR-2=           9,98000000
TEST CASE 2
----------
OVAR-3 =           0,00000997
OVAR-4 =           9,97000000
TEST CASE 3
----------
OVAR-5 =           0,00000998
OVAR-6 =           9,98000000
TEST CASE 4
----------
OVAR-7 =           0,00000997
OVAR-8 =           9,97000000
TEST CASE 5
----------
OVAR-9 =           0,00000998
OVAR-10=           9,98000000
TEST CASE 6
----------
OVAR-11=           0,00000997
OVAR-12=           9,97000000
----------
FACTOR-1 =           6,00000000
FACTOR-2 = 6
FACTOR-3 = 6,0
FACTOR-4 =     6
```

Rolf Parker
Systems Programmer (Germany)

IPLing just got easier

The questions began when it was discovered that SYS1.IPLPARM had over 100 members. Eighteen were currently in use in regular rotation, as IODF and IOCDS were updated. And the operators had a regularly-updated wall chart with the current and previous IPL parameter value for each of the six LPARs on a single zSeries 900.

With a little help from another new person on-site, SYS1.IPLPARM is now down to one member. Meanwhile, the operators can forget about the IPL parameter value because it can be set once on the HMC and left: it remains constant across all LPARs, even when the IODF changes.

WHERE WE STARTED

The main production system, MVSA, had been using SYS1.IPLPARM member LOAD2A:

```
IODF     Ø2 SYS1     EPRDMVSA ØØ                              NUCLEUS  1
SYSCAT   ASYSL1113CSYS1.PRODPLEX.CATALOG
SYSPARM  (ØØ,SA)
IEASYM   (ØØ,RS)
INITSQA  ØØØØM ØØØ1M
NUCLST   ØØ N
PARMLIB  SYS1.SYSRES.EXTNSION.PARMLIB                  ******
PARMLIB  SYS1.ZOS12.PARMLIB
PARMLIB  SYS1.PARMLIB
```

MVSB, the system used to install and test new software, had been using member LOAD2Z:

```
IODF     Ø2 SYS1     ATSTMVSB ØØ                              NUCLEUS  1
SYSCAT   BSYSL1113CSYS1.MVSB.CATALOG
SYSPARM  (SZ,SB)
NUCLST   ØØ N
IEASYM   (ØØ,RS)
INITSQA  ØØØØM ØØØ1M
PARMLIB  SYS1.SYSRES.EXTNSION.PARMLIB                  ******
PARMLIB  SYS1.ZOS12.PARMLIB
PARMLIB  SYS1.PARMLIB
```

As you can see, the LOAD*xx* members are similar, but not identical. The most obvious difference is on the first line, where the operating

system configuration identifier differs for each LPAR – ATSTMVSB versus EPRDMVSA. LOAD1A (not shown) differs from LOAD2A (see above) only by the IODF number specified just after IODF on the first line. If you rotate between three IODFs, you must have three LOAD*xx* members. We have six LPARs, so that is how we got to 18 active LOAD*xx* members.

NEW Z/OS FEATURES

Two features have been added to z/OS in recent years that make it possible to use one LOAD*xx* member instead of 18. Eliminating the specification of the IODF number on the IODF statement is possible with HCD, because it can insert an IODF pointer into the IOCDS. Unfortunately, it does not happen by default because HCD releases the IOCDS build job while it still has the IODF file open. To overcome this problem, be sure to specify TYPRUN=HOLD on the JOB card that you create for HCD just before submitting the IOCDS build batch job. Exit HCD and then release the job.

Instead of specifying the IODF number, there are a number of different symbols you can specify. I recommend equals signs (==) because a Wait state is forced if the IODF pointed to by the IOCDS does not exist.

The second new feature is the ability to specify statements specific to each LPAR in a single LOAD*xx* member. The LPARNAME statement provides this capability.

JUST ONE IPLPARM MEMBER

Put it all together and here is what you get as LOAD00, which works for all IODFs and all LPARs:

```
NUCLEUS  1                                                  SYSCAT
ASYSL1113CSYS1.PRODPLEX.CATALOG
IEASYM   (ØØ,RS)
INITSQA  ØØØØM ØØØ1M
NUCLST   ØØ N
PARMLIB  SYS1.SYSRES.EXTNSION.PARMLIB                ******
PARMLIB  SYS1.ZOS12.PARMLIB
PARMLIB  SYS1.PARMLIB
LPARNAME EPRDMVSA
```

```
IODF     == SYS1     EPRDMVSA ØØ
SYSPARM  (ØØ,SA)
LPARNAME APRDMVSE
IODF     == SYS1     APRDMVSE ØØ
SYSPARM  (ØØ,SE)
LPARNAME CPRDMVSH
IODF     == SYS1     CPRDMVSH ØØ
SYSPARM  (ØØ,SH)
LPARNAME APRDMVSI
IODF     == SYS1     APRDMVSI ØØ
SYSPARM  (ØØ,L)
SYSCAT   ISYSL1113CSYS1.MVSI.CATALOG
LPARNAME ATSTMVSB
IODF     == SYS1     ATSTMVSB ØØ
SYSPARM  (SZ,SB)
SYSCAT   BSYSL1113CSYS1.MVSB.CATALOG
```

The statements common to all LPARs are included first, then those for each LPAR. Note how statements that can be specified once, such as SYSCAT, can be overridden for a specific LPAR. Finally, if you are wondering about the six asterisks to the far right of the first PARMLIB in all the LOAD*xx* members shown, that indicates that the parmlib is located on the system residence volume; change system residence volumes and you are using a different but identically-named parmlib.

DISPLAY IPLINFO

Not sure what LOAD*xx* member or IODF is in use? The console command Display IPLINFO provides a lot of useful information. On MVSB, with the new LOAD00 in place, here is what you would see:

```
IEA63ØI  OPERATOR E667800  NOW ACTIVE,   SYSTEM=MVSB    , LU=N11521A D
IPLINFO
IEE254I  15.19.27 IPLINFO DISPLAY Ø31
 SYSTEM IPLED AT Ø6.35.58 ON Ø4/11/2ØØ3
 RELEASE z/OS   Ø1.Ø2.ØØ
 USED LOADØØ IN SYS1.IPLPARM ON 45DF
 ARCHLVL = 2   MTLSHARE = N
 IEASYM LIST = (ØØ,RS)
 IEASYS LIST = (SZ,SB) (OP)
 IODF DEVICE 45DF
 IPL DEVICE 491B VOLUME OS39ØM
```

Before the change, here is what it looked like on MVSA:

```
D IPLINFO                                IEE254I  15.22.Ø3 IPLINFO
```

```
DISPLAY 921
 SYSTEM IPLED AT 16.26.18 ON Ø4/13/2ØØ3
 RELEASE z/OS   Ø1.Ø2.ØØ
 USED LOAD2A IN SYS1.IPLPARM ON 45DF
 ARCHLVL = 2   MTLSHARE = N
 IEASYM LIST = (ØØ,RS)
 IEASYS LIST = (ØØ,SA) (OP)
 IODF DEVICE 45DF
 IPL DEVICE 4Ø2A VOLUME ZOSØØ2
```

THE CHANGE REQUEST

Implementation of this change – having just one member (LOAD00) in SYS1.IPLPARM – required a change request. Here are the implementation and backout instructions given to data centre staff to make this change during a scheduled IML.

Implement

Set IOCDS – Change IOCDS to A0.

On the HMC, double-click *Groups* then double-click *Defined CPCs* then double-click the *A20641C4* icon. Unlock (set Lockout Disruptive Tasks to No) and then hit the *Change Options* button. Select PORA0 as the Profile Name (from the list). Push the two *Save* buttons that appear.

Activate

On the HMC, single-click on the A20641C4 icon, and double-click on the *Activate* icon. (To see the Activate icon, you may have to repeatedly click on the *Rotate* icon (circular arrow) in the bottom-right corner of the screen.)

IPL each system with the new standard IPL parameters – 45DF00 for all systems.

Backout

Set IOCDS – Change IOCDS to A2.

On the HMC, for A20641C4, change the profile to DEFAULT (see Implementation Plan for details) and Activate.

Re-IML.

Re-IPL each system with the previous IPL parameters:

MVSA – 45DF2A

MVSE – 45DF2E

MVSH – 45DF2H

MVSI – 45DF2i.

OTHER CHANGES

Another improvement was implemented at the same time. As shown in the first step of *Implement*, the PORA0 IML profile had been previously created with A0 specified as the IOCDS. Previously, the operators had to change the IOCDS number in the DEFAULT IMS profile. PORA1 and PORA2 were also created for future IMLs when the IOCDS is changed.

The final change made was to keep the IOCDS and IODF numbers in sync. Now, IOCDS A0 is always used with IODF00, A1 with IODF01, and A2 with IODF02. A three-way rotation seemed more than adequate.

NEXT

What is next? We hope to substantially reduce the number of LPARs. Historically, they had been created to resolve performance problems. But those are now being addressed with a complete, from-scratch, redesign of WLM settings. Preliminary results are very encouraging.

Both Cheryl Watson and the Washington System Centre (WSC) provide starting points for WLM settings. WSC was chosen, with test results reviewed by SHARE speaker Tom Russell of IBM, and his suggested improvements implemented.

Who knows? We may even be able to repeat last month's processor downgrade, and move down yet another level of cost.

Jon E Pearkins
Adiant Corporation (Canada)

Using TAR and JAR files on MVS

I recently needed to transmit a large number of TIF images (held in MVS datasets) to a server running a Windows environment. The Unix TAR command can be used under Unix System Services on MVS to create an archive file that can then be transmitted to the server and expanded using the familiar Windows ZIP utility.

In the following example the images are first copied from a PDS to an HFS directory using the OPUTX EXEC.

```
//*
//* use the OPUTX exec to copy each member of the
//* PDS into the HFS directory
//*
//OPUTX EXEC PGM=IKJEFT01
//SYSEXEC  DD DSN=SYS1.SBPXEXEC,DISP=SHR
//SYSPROC  DD DISP=SHR,DSN=SYS1.SISPCLIB
//ISPPLIB  DD DISP=SHR,DSN=SYS1.SISPPENU
//         DD DISP=SHR,DSN=SYS1.SISFPLIB
//ISPMLIB  DD DISP=SHR,DSN=SYS1.SISPMENU
//         DD DISP=SHR,DSN=SYS1.SISFMLIB
//         DD DISP=SHR,DSN=SYS1.SBPXMENU
//ISPSLIB  DD DISP=SHR,DSN=SYS1.SISPSENU
//         DD DISP=SHR,DSN=SYS1.SISPSLIB
//ISPTLIB  DD DISP=SHR,DSN=SYS1.SISPTENU
//         DD DISP=SHR,DSN=SYS1.SISFTLIB
//ISPPROF   DD DSN=&&ISPPROF,SPACE=(TRK,(5,1,2)),LIKE=&userid.ISPPROF,
//         DISP=(NEW,PASS)
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
  OPUTX 'your-PDS-of-images' '/your-dir' ASIS BINARY CONVERT(NO) +
        MODE(644) SUFFIX(tif)
/*
```

The TAR command is then used to create the archive dataset – note that in this particular case the compression flag is not used because TIF images are already in a compressed format.

```
//*
//* use Unix System Services TAR command
//* to create an archive dataset
//*
//TAR    EXEC PGM=BPXBATCH,
//         PARM='sh tar -cUvf //tiffs.tar /your-dir'
//STDOUT  DD PATH='/tmp/stdout',
```

```
//           PATHOPTS=(OCREAT,OWRONLY),PATHMODE=SIRWXU,
//           PATHDISP=KEEP
//STDERT   DD PATH='/tmp/stderr',
//           PATHOPTS=(OCREAT,OWRONLY),PATHMODE=SIRWXU,
//           PATHDISP=KEEP
//*
```

The resulting MVS dataset, &userid.TIFFS.TAR, can be sent (obviously as a binary transmit) to the server and unzipped to restore the individual image files. Note that the Windows ZIP program will recognize files with an extension of .tar.

If you want to include text files in your archive, there is a slight problem in that we are using a binary transfer – somewhere along the line the data will need to be translated into ASCII. The following job will take an MVS text file (ie EBCDIC character data) and convert it to ASCII. The TEXT option in this case will result in the addition of an LF character being added to delimit the end of each record. This will be OK for a Unix system; however, the standard Windows text file expects CR LF or X'0D0A' as the delimiter.

```
//OCOPY  EXEC PGM=IKJEFT01
//INDD     DD DSN=your-text-file,DISP=OLD
//OUTHFS   DD PATH='/your-dir/myfile.txt',
//            PATHDISP=(KEEP,DELETE),
//            PATHOPTS=(OWRONLY,OCREAT),PATHMODE=SIRWXU
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
  OCOPY INDD(INDD) OUTDD(OUTHFS) TEXT CONVERT((BPXFX311)) +
        FROM1047
/*
```

While the Windows ZIP utility is quite happy to work with Unix TAR files created on MVS, the reverse is not true. To cater for the situation where you may want to package up some files on a Windows environment and upload them to MVS as a single dataset, you can make use of the Java archive utility, which provides a similar ZIP functionality, and, being Java, provides platform-independence. Further information about Java archive files can be found on the Internet (see the Sun Java tutorial pages).

Assuming that you have made Java available on your MVS system (mounting the HFS containing the Java software and setting the environment variables), the following JCL illustrates the use of the JAR command to expand a Java archive file that was created on a

Windows PC:

```
//* use JAR to unload the archive
//*
//JAR    EXEC PGM=BPXBATCH,
//    PARM='sh cd work;jar xf /work/fixes.jar'
//STDOUT  DD PATH='/tmp/stdout',
//           PATHOPTS=(OCREAT,OWRONLY),PATHMODE=SIRWXU,
//           PATHDISP=KEEP
//STDERT  DD PATH='/tmp/stderr',
//           PATHOPTS=(OCREAT,OWRONLY),PATHMODE=SIRWXU,
//           PATHDISP=KEEP
//*
```

The OGETX EXEC can then be used to move files into a standard MVS PDS:

```
OGETX /work '&userid.MVS.PDS' LC SUFFIX(bin) BINARY
```

With many vendors providing fixes over the Internet as downloads, one possible use for this kind of job would be to move a number of fixes from a directory on your PC up to an MVS dataset.

Dave Welch (New Zealand)

Analysing data-in-virtual statistics

Since its introduction a long time ago (with MVS/XA!), when it received some attention, DIV (Data-In-Virtual) seems to have fallen into oblivion. The main reason for that is the fact that DIV is somewhat difficult to use because the Assembly-language primitive functions one must use are not readily available in high-level languages. However, DIV, which is a set of primitive functions, enables an application program to load and manage substantial amounts of data into memory from a VSAM Linear DataSet (LDS). The LDS itself can grow to 4GB and the program can map up to (almost) 2GB of it at a time in central memory . Applications can create, read, and update data without the I/O buffer, blocksize, and record considerations that the traditional

GET and PUT types of access method require. An application written for data-in-virtual views its permanent storage data as a seamless body of data without internal record boundaries. Among the applications that can be considered for a data-in-virtual implementation are applications that process large arrays, VSAM relative record applications, and BDAM fixed-length record applications. The potential benefits may be realized eventually as DIV merges with hiperspaces (a related concept) and as subsystems, languages, and application packages exploit the DIV benefits. For example, DIV is used by DFSMS when I/O to SMS control datasets is needed.

The data-in-virtual services process the application data in 4096-byte (4KB) units on 4KB boundaries called blocks. The application data resides in what is called a data-in-virtual object, a data object, or simply an object. The data-in-virtual object is a continuous string of uninterrupted data. When one writes an application using the techniques of data-in-virtual, the I/O takes place only for the data referenced and saved. Only the referenced pages of such an object are brought into virtual storage. Bytes of the mapped pages can be accessed and changed in normal program execution without regard to the need for updating. On request, or at the time the connection to the DIV object is terminated, only the changed pages are written back to the linear dataset. If one runs an application using conventional access methods, and then runs it again using data-in-virtual, one will notice a difference in performance. This difference depends on both the size of the dataset and its access pattern. To gain the right to view or update the object, an application must use the ACCESS service. ACCESS is similar to the OPEN macro of VSAM. It has a mode parameter of READ or UPDATE, and it gives your application the right to read or update the object. If the application has finished processing the object, it uses UNACCESS to relinquish access to the object.

Before using the DIV macro to process a linear dataset object (or a hiperspace object), one must create the dataset (or the hiperspace). *OS/390 MVS Programming:Authorized Assembler Services Guide (*GC28-1763) explains how to use DIV macro functions. The ‘how to’ reference for hiperspaces is the *Extended Addressability Guide,*

(GC28-1468). Also, it is worth consulting *An Introduction to Data-in-Virtual* (GG66-0259), which may be a bit old but provides a few Assembler, Fortran, and PL/I examples.

COLLECTING DIV DATA

When enabled by the SMFPRM*xx* TYPE parameter, SMF creates record type 41, which provides resource usage information regarding data-in-virtual. There are two subtypes of this SMF record: subtype 1 is an ACCESS record – the ACCESS data section is written when a DIV object is accessed; subtype 2 is an UNACCESS record – the counts for the I/O activity section are accumulated by data-in-virtual while the object is in use and are reported at the time of the UNACCESS request. The subtype 2 record is written whenever a data-in-virtual object is unaccessed.

A detailed description of layout of SMF type 41 record and its subtypes can be obtained from the *MVS System Management Facilities (SMF)* (SA22-7630-03) manual. You can also find the subtype descriptions in the macro ITVSMF41 in SYS1. MACLIB.

Based on record descriptions obtained from above mentioned manual a simple DIV report writer was written.

CODE

The code is a two part stream. In the first part (COPY412), selected SMF records (selection being defined by INCLUDEs condition) are copied from an SMF dataset to a VB file, which is the input file for the second step.

In the second step (DIV412), the captured records are formatted by invoking REXX EXEC (DIV412), and a report produced. The report shows the users performing access/unaccess of an object, along with the timestamp of when the object was accessed/ unaccessed, the size of the object, and its read/write/re-read count data. Elapsed time during which the object was read or updated is calculated from two TOD timestamps.

JCL

```
//DIVJOB    JOB ACCT#,
//             MSGLEVEL=(1,1),
//             MSGCLASS=R,
//             NOTIFY=&SYSUID
//COPY412  EXEC PGM=ICETOOL
//TOOLMSG  DD SYSOUT=*
//DFSMSG   DD SYSOUT=*
//RAWSMF   DD DISP=SHR,DISP=hlq.SMFDUMPW
//SMF412   DD DSN=your.copied.by.sort.to.VB.smf.dataset,
//         SPACE=(CYL,(1)),UNIT=SYSDA,DISP=(NEW,PASS),
//         DCB=(RECFM=VB,LRECL=32756,BLKSIZE=3276Ø)
//TOOLIN   DD *
  COPY FROM(RAWSMF) TO(SMF412) USING(SMFI)
//SMFICNTL DD *
  OPTION SPANINC=RC4,VLSHRT
  INCLUDE COND=(6,1,BI,EQ,41,AND,23,2,BI,EQ,2)
/*
//DIV412  EXEC PGM=IKJEFTØ1,REGION=ØM,DYNAMNBR=5Ø,PARM='%DIV412'
//SYSEXEC  DD DISP=SHR,DSN=your.rexx.library
//SMF      DD DISP=(SHR,PASS),DSN=your.copied.by.sort.to.VB.smf.dataset,
//DIV41    DD DSN=your.div.report.dataset,
//         SPACE=(CYL,(1,1)),UNIT=SYSDA,
//         DISP=(NEW,KEEP),DCB=(RECFM=FB,LRECL=15Ø)
//SYSPRINT DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD DUMMY
/*
```

DIV412 EXEC

```
/* REXX EXEC to read and format SMF record 41.2*/
ADDRESS TSO
/*-------------------------------------------------------------------*/
/* Print report header and labels                                    */
/*-------------------------------------------------------------------*/
 Out.1 = left(' ',3Ø,' '),
         ||center('Data in Virtual Report ',22,),
         ||left(' ',15,' ')
 Out.2 = left(' ',2Ø,' '),
         ||center('Report produced on',18,),
         ||left(' ',1,' ')||left(date(),11),
         ||left(' ',1,' ')||left('at ',3,' '),
         ||left(time(),1Ø)
 Out.3 = '  '
 Out.4 = left('SMF date',11) left('SMF time',9),
         left('SID',4)       left('DFP lvl.',7),
         left('Job name',9)  left('DD name',9),
         left('Access Time',15) left('Unaccess Time',15),
```

```
        left('A.Size',6)    left('U.Size',7),
        left('Mode',5)      left('Elaps.sec.',1Ø),
        left('Block',5)     left('Block',6),
        left('Block',6)     left('Read',5),
        left('Write',5)
Out.5 = left(' ',118)       left('read',5),
        left('write',6)     left('rread',6),
        left('I/O',5)       left('I/O',3)
Out.6 = LEFT('-',149,'-')
"EXECIO * DISKW DIV41 (STEM Out.)"
'EXECIO * DISKR SMF ( STEM x. FINIS'
   do i = 1 to x.Ø
 /*-------------------------------------------------------------*/
 /*           Header/Self-defining Section                      */
 /*-------------------------------------------------------------*/
 smftype  = c2d(SUBSTR(x.i,2,1))             /* SMF record type */
 smfstype = c2d(SUBSTR(x.i,19,2))            /* Record subtype  */
 IF smftype = '41'  Then
 Do
  smfdate = SUBSTR(c2x(SUBSTR(x.i,7,4)),3,5) /* Unpack SMF date */
  smftime = smf(c2d(SUBSTR(x.i,3,4)))        /* Decode SMF time */
  sysid   = SUBSTR(x.i,11,4)           /* System identification */
    trp   = c2d(SUBSTR(x.i,21,2))   /*        number of triplets*/
    opd   = c2d(SUBSTR(x.i,25,4))   /* offset to product section*/
    lpd   = c2d(SUBSTR(x.i,29,2))   /* length of product section*/
    npd   = c2d(SUBSTR(x.i,31,2))   /*number of product sections*/
    od1   = c2d(SUBSTR(x.i,33,4))   /*  offset to access section*/
    ld1   = c2d(SUBSTR(x.i,37,2))   /*  length of access section*/
    nd1   = c2d(SUBSTR(x.i,39,2))   /* number of access sections*/
    od2   = c2d(SUBSTR(x.i,41,4))   /*offset to unaccess section*/
    ld2   = c2d(SUBSTR(x.i,45,2))   /*length of unaccess section*/
    nd2   = c2d(SUBSTR(x.i,47,2))  /*number of unaccess sections*/
    od3   = c2d(SUBSTR(x.i,49,4))   /*    offset to i/o activity*/
    ld3   = c2d(SUBSTR(x.i,53,2))   /*    length of i/o activity*/
    nd3   = c2d(SUBSTR(x.i,55,2))   /*    number of i/o activity*/
 /*-------------------------------------------------------------*/
 /*           Product Section                                   */
 /*-------------------------------------------------------------*/
   IF opd <> Ø AND npd <> Ø Then do
      opd=opd-3
      dfplvl =  SUBSTR(x.i,opd,8)            /*   product level */
      prod   =  SUBSTR(x.i,opd+8,16)         /*  component name */
   end
 /*-------------------------------------------------------------*/
 /*           Object ACCESS Data Section                        */
 /*-------------------------------------------------------------*/
   IF od1 <> Ø and nd1 <> Ø Then do
      od1=od1-3
      dda =  SUBSTR(x.i,od1,8)                  /*  object ddname*/
      aza =  c2d(SUBSTR(x.i,od1+8,4))           /*  object size  */
      ata=   SUBSTR(ct(c2x(SUBSTR(x.i,od1+12,4))),11,15) /* TOD  */
```

```
       tya =  c2d(SUBSTR(x.i,od1+16,1))          /*  object type  */
       ama =  c2d(SUBSTR(x.i,od1+17,1))          /*  access mode  */
       jbn =  SUBSTR(x.i,od1+18,8)        /* jobname/started task */
           SELECT
             when ama=1 then mode='Read'
             when ama=2 then mode='Update'
           END
      end
   /*-------------------------------------------------------------*/
   /*           Object UNACCESS Data Section                      */
   /*-------------------------------------------------------------*/
     IF od2 <> 0 and nd2 <> and then do
        od2=od2-3
        uzu =  c2d(SUBSTR(x.i,od2,4))               /* object size*/
        utu =  SUBSTR(ct(c2x(SUBSTR(x.i,od2+4,4))),11,15)   /* TOD*/
       end
   /*-------------------------------------------------------------*/
   /*           Object I/O Activity Section                       */
   /*-------------------------------------------------------------*/
     IF od3 <> 0 and nd3 <> 0 then do
        od3=od3-3
        brd =  c2d(SUBSTR(x.i,od3,4))       /*    tot.no. of reads*/
        bwr =  c2d(SUBSTR(x.i,od3+4,4))     /*   tot.no. of writes*/
        brr =  c2d(SUBSTR(x.i,od3+8,4))     /* tot.no. of re-reads*/
        inc =  c2d(SUBSTR(x.i,od3+12,4))  /*tot.no.of i/o for read*/
        ouc =  c2d(SUBSTR(x.i,od3+16,4)) /*tot.no.of i/o for write*/
     end
     timedif=dif(utu,ata)      /* how long the object was accessed*/
   /*-------------------------------------------------------------*/
   /* formatting and printing a DIV entry                         */
   /*-------------------------------------------------------------*/
   divout = left(Date('N',smfdate,'J'),11) left(smftime,9),
            left(sysid,4)  left(dfplvl,8)  left(jbn,8),
            left(dda,8)    right(ata,15)   left(utu,17),
            left(aza,6)    left(uzu,6)     left(mode,7),
            right(timedif,8),
            right(brd,4)   right(bwr,5)    right(brr,5),
            right(inc,5)   right(ouc,5)
    PUSH divout
      "EXECIO 1 DISKW DIV41"
      end
    end
exit
```

SMF PROCEDURE

```
/* REXX - convert a SMF time  */
arg time
    time1    = time % 100
    hh       = time1 % 3600
    hh       = RIGHT("0"||hh,2)
```

```
    mm       = (time1 % 60) - (hh * 60)
    mm       = RIGHT("0"||mm,2)
    ss       = time1 - (hh * 3600) - (mm * 60)
    ss       = RIGHT("0"||ss,2)
    otime  = hh||":"||mm||":"||ss          /* Compose SMF time*/
    return otime
```

CT PROCEDURE

```
/*                                                  */
/* TOD timestamp is a 16-byte EBCDIC representat */
/* The BLSUXTOD proc is described in "z/OS       */
/* V1R3 MVS IPCS Customization"                    */
/*                                                  */
arg todtime
If todtime    <> '0000000000000000' Then
  Do
    TOD_Value = X2C(todtime)
    Returned_Date = '--------------------------'
    address LINKPGM "BLSUXTOD TOD_Value Returned_Date"
  End
Else
  Returned_Date = ''
Return Returned_Date
```

DIF PROCEDURE

```
/*                                                  */
/* Dif: REXX subroutine to find the              */
/* difference between two timestamps, in this    */
/* case in seconds                                  */
/*                                                  */
arg time2,time1
 parse var time2 h2 ':' m2 ':' s2 ':' t2
 parse var time1 h1 ':' m1 ':' s1 ':' t1
 tot2 = h2*3600 + m2*60 + s2
 tot1 = h1*3600 + m1*60 + s1
 es = tot2 -tot1
 if es<0 then es=es+86400
 eh=es%3600
 es=es//3600
 ex=es
 em=es%60
 es=es//60
/* et=right(eh,2,0)':'right(em,2,0)':'right(es,2,0) */
 return ex
```

Mile Pekic
Systems Programmer (Serbia and Montenegro)

Format and display a data field from Assembler

BACKGROUND

Often during testing and for one-off applications, it is useful to have an easy means of displaying, from an Assembler program, the contents of a field, converted to a displayable format when necessary (such as for binary or hexadecimal fields) – ie something similar to the COBOL DISPLAY instruction or the C printf() function.

SOLUTION

The DISPLAY macro described in this article outputs the contents of a specified field to the job log (routing code 11). The use of the WTO macro obviates the need to specify a DD statement and also has 31-bit capability. Furthermore, it can be used in two popular environments – batch and TSO.

To reduce the footprint of the generated code when the macro is used more than once in a program, the code used to perform the formatting and output is included just once. Similarly, the use of sparse translation tables (not all 256 bytes defined) reduces the size, but means that the first macro call should not be placed too near the start of the program, otherwise addressing errors may occur (if necessary, the appropriate padding must be included).

To improve the utility, the field name is also output (not for literals).

RUN-TIME ENVIRONMENT

The DISPLAY macro can run in batch and TSO.

Note: the macro could easily be extended to run in other environments, for example, CICS.

The invocation syntax is:

```
[name] DISPLAY source[,length[,type]]
```

where:

- *name* – optional label. The label applies to the source.
- *source* – source field; field name (eg ALPHA), literal (eg 'beta gamma') or base-displacement address (eg 4(5) = 4 byte displacement from the address contained in general purpose register 5).
- *length* – explicit length (in bytes) of the source field; either as a self-defining (numeric) value (eg 8) or as a register (specified within parentheses, eg (9)) that contains the appropriate length at execution time. The length must be specified for a base-displacement address. N = numeric (=decimal) value for a register, otherwise the register content is displayed in hexadecimal notation. If no length is specified, the implicit length is used, ie the value returned by the L attribute.
- *type* – field type. If no type is specified, the implicit type is used, ie the value returned by the T attribute. Type may be one of the following:

  C – character

  Z – zoned (decimal)

  X – hexadecimal

  P – packed decimal (signed)

  B, H, F – binary (signed)

  A – address

  R – general purpose register (0,…,15 or appropriate equate specified as *field*).

Register usage: as usual for macros, DISPLAY uses registers 14-1.

MACRO DEFINITION

```
         MACRO
&NAME    DISPLAY &P1,&LP1,&TP1
.**
.* Format and display a data field
```

```
.**
.*  Parameters:
.*   P(1) - source field start (or literal)
.*   P(2) - source field length, either numeric literal or register (n)
.*          (if omitted, default length used)
.*          ('N' = numeric (decimal) conversion for register)
.*   P(3) - source field type (optional)
.**
.* The following field types are supported:
.*  C - character
.*  Z - zoned (decimal)
.*  X - hexadecimal
.*  P - packed decimal (signed)
.*  B, H, F - binary (signed)
.*  A - address
.*  R - register
.*  Literal (field enclosed within quotes)
.*  Explicit address (e.g. Ø(R1)), length must be specified
.**
         GBLB  &FD
         LCLA  &L
         LCLC  &LN
         LCLC  &C,&W,&MK
.* label
         AIF   (T'&NAME EQ 'O').AØ
&NAME    DS    ØH
.AØ      ANOP
         MVC   ##WK,##WK-1 clear
.* 1st CALL?
         AIF   (&FD).A1
&FD      SETB  1
         B     ##GO1
         SPACE 1
##FD     DS    PL8
##MK1    DC    X'Ø1Ø3Ø7ØF'
##MK2    DC    X'2Ø4Ø7Ø9Ø'
         SPACE 1
         DC    C' '
##OUT    DS    ØCL8Ø
##LEN    DS    HL2
##NAME   DS    CL8
         SPACE 1
         DC    C' '
##WK     DS    ØCL71
##WKS    DC    C' ' SIGN
##WKFLD  DS    CL7Ø
         SPACE 1
##FTR    DC    CL16'Ø123456789ABCDEF'
         SPACE 2
##GO1    DS    ØH
```

```
.A1      MVC   ##OUT,##OUT-1
.*
&TP      SETC  T'&P1
         AIF   (T'&TP1 EQ 'O').A1A
&TP      SETC  '&TP1'
.A1A     ANOP
.*
         SPACE 1
&C       SETC  '&P1'(1,1)
         AIF   ('&C' EQ '''').B4
&L       SETA  K'&P1
         MVC   ##NAME,=CL8'&P1'
         AIF   ('&TP' EQ 'H').B7
         AIF   ('&TP' EQ 'F').B7
         AIF   ('&TP' NE 'B').B1
.B7      ANOP
.* binary
&LN      SETC  'L''&P1-1'
         AIF   (T'&LP1 EQ 'O').B7A
&LN      SETC  '&LP1-1'
.B7A     SR    Ø,Ø
         LA    1,&LN
         IC    1,##MK1(1)
         ICM   Ø,Ø,&P1
         EX    1,*-4
         CVD   Ø,##FD
         LA    1,&LN
         IC    1,##MK2(1)
         AGO   .A2
.B1      AIF   ('&TP' NE 'P').B5
.* packed decimal
         ZAP   ##FD,&P1
&LN      SETC  'L''&P1*2-1'
         AIF   (T'&LP1 EQ 'O').B1A
&LN      SETC  '&LP1*2-1'
.B1A     LA    1,(&LN)*16
.A2      MVI   ##WKS,C'+'
         CP    ##FD,=P'Ø'
         BNL   *+8
         MVI   ##WKS,C'-'
         OI    ##FD+7,X'ØF'
         UNPK  ##WKFLD(Ø),##FD
         EX    1,*-6
         AGO   .MPUT
.B5      ANOP
         AIF   ('&TP' EQ 'C').B5B
         AIF   ('&TP' NE 'Z').B6
.B5B     ANOP
.* character or zoned decimal
&LN      SETC  'L''&P1'
```

```
         AIF   (T'&LP1 EQ 'O').B5A
&C       SETC  '&LP1'(1,1)
         AIF   ('&C' NE '(').B5D
         LR    1,&LP1
         BCTR  1,Ø
         AGO   .B5C
.B5D     ANOP
&LN      SETC  '&LP1'
.B5A     LA    1,&LN-1
.B5C     LA    Ø,(L'##WKFLD-1)
         CR    1,Ø
         BNH   *+6
         LR    1,Ø
         MVC   ##WKFLD(Ø),&P1
         EX    1,*-6
         AGO   .MPUT
.B6      AIF   ('&TP' NE 'X').B8
.* hexadecimal
.B6C     ANOP
&LN      SETC  'L''&P1'
&P       SETC  '&P1'
         AIF   (T'&LP1 EQ 'O').B6A
&C       SETC  '&LP1'(1,1)
         AIF   ('&C' NE '(').B6D
         LR    Ø,&LP1
         AGO   .B6E
.B6D     ANOP
&LN      SETC  '&LP1'
.B6A     LA    Ø,&LN
.B6E     LA    1,(L'##WKFLD/2)
         CR    Ø,1
         BNH   *+6
         LR    Ø,1
         LA    1,&P
.B6B     LA    15,##WKFLD
         UNPK  ##FD(3),Ø(2,1)
         TR    ##FD(2),##FTR-X'FØ'
         MVC   Ø(2,15),##FD
         LA    1,1(1)
         LA    15,2(15)
         BCT   Ø,*-26
         AGO   .MPUT
.B4      ANOP
.* literal
&L       SETA  K'&P1-2
         MVC   ##WKFLD(&L),=C&P1
         AGO   .MPUT
.MPUT    SPACE
         MVC   ##LEN,=AL2(L'##WK)
         WTO   TEXT=((##OUT,D)),ROUTCDE=11
```

```
         MEXIT
.B8      AIF   ('&TP' NE 'R').B9
.* register
         AIF   ('&LP1' EQ 'N').B8A    decimal
         ST    &P1,##FD+4
         LA    Ø,4
         LA    1,##FD+4
         AGO   .B6B
.*
.B8A     CVD   &P1,##FD
         LA    1,11*16
         AGO   .A2
.*
.B9      AIF   ('&TP' NE 'A').B1Ø
.* ADDRESS
         MVC   ##FD+4(4),&P1
         AGO   .B6C
.*
.B1Ø     AIF   ('&TP' NE 'U').E1
         AIF   ('&C' LT 'Ø').E1
.* explicit address
.* type hexadecimal (implicit)
         LA    15,&P1
&P       SETC  'Ø(15)'
         AGO   .B6D
.*
.E1      MNOTE 8,'*** INVALID DATA TYPE ***'
         MEXIT
.E2      MNOTE 8,'*** INVALID LENGTH ***'
         MEND
```

SAMPLE CODE FRAGMENT

```
…
         LA      15,2Ø
         DISPLAY 15,,R      R15 (hex)
         LA      15,2Ø
         DISPLAY 15,N,R     R15 (decimal)
         DISPLAY 'tag'      literal
         DISPLAY PID
         DISPLAY FNO,1,X
         DISPLAY FDATA,8,C
         LA      2,TEXT     set base address
         DISPLAY 5(2),4,C
         DISPLAY CTR        packed decimal
         DISPLAY ZCTR       zoned decimal (with sign)
         LA      2,4        data length
         DISPLAY text,(2)   truncate
         DISPLAY text       complete field
```

```
…
PID      DC    CL4'1234'
FLD      DS    ØCL256
FNO      DC    AL1(8)
FLEN     DC    AL1(16)
FDATA    DC    CL254'alpha'
TEXT     DC    C'beta gamma'
CTR      DC    P'-79'
```

ASSOCIATED OUTPUT

```
15        00000014
15       +000000000020
          tag
PID       1234
FNO       Ø8
FDATA     alpha
5(2)      gamm
CTR      -0079
ZCTR      7H
text      beta
text      beta gamma
```

A Rudd
Systems Programmer (Germany)

Researching CHPID problems

CHPID problems can point to serious I/O problems that can affect DASD, tape, or communication devices. There are many messages that can identify CHPID problems. This article was originally written for operations and shows how to determine whether a CHPID problem is a major or minor concern.

WHAT IS A CHPID?

A CHPID is a Channel Path ID. MVS has always had the ability to use channels, control units, and devices to accomplish input/output (I/O) operations. A device (like DASD, tape, printers, etc) is always represented in the operating system as a Unit Control Block (UCB). Devices are connected to control units and control units are attached

to the mainframe with channels. The pre-MVS/XA naming convention for UCBs enforced a three-digit numbering scheme and was made up of the one-digit channel, plus a one-digit control unit, plus a one-digit device number (eg A26 – channel A, control unit 2, device number 6). The hardware and software architecture allowed for only 4,096 I/O devices per mainframe. When MVS evolved to MVS/XA (early 1980s), the I/O subsystem was enhanced to allow for more than 256 devices per channel and up to eight paths to each device. With MVS/XA, the I/O subsystem was significantly enhanced and the ability to use four-digit UCBs allowed the addition of over 65,000 I/O devices. The old naming conventions were abandoned and the introduction of a new logical mapping of a physical channel to a logical path was now necessary. Hence the creation of Channel Path IDs, or CHPIDs, to help us exploit the more powerful I/O subsystem. So a CHPID is a logical path from a device to a physical channel. With current control unit technology, each device can have up to eight physical paths to perform I/O.

WHAT DOES A CHPID FAILURE MEAN?

A CHPID failure means a physical channel has failed or had a severe problem. Since most channels these days are ESCON or FICON, a failure is usually associated with a 'loss of light'. If there are many devices on this channel, it may be a major problem. If there are only a few devices on the channel, or if all the devices on the channel have multiple alternative paths through unaffected channels, this may not be a major problem. Since each channel can support multiple devices and each device can 'ride' multiple channels, it is necessary to know what devices are on which channels.

HOW DO WE KNOW WE HAVE A CHPID PROBLEM?

The most likely indication of a CHPID problem will be a message on the console or an automation alert. Occasionally, the CEC will 'phone home' with a CHPID problem and IBM will call. If the IBM Support Center calls to report a problem, we will usually have seen an alert for the CHPID error and problem determination should already be in progress. IBM will usually tell us which CEC reported

the problem. The IBM Support Center does not know our CECs' names; they will give us the IBM serial number for the box. Always match serial numbers to CECs to determine the affected LPARs. Armed with this information, always check to see whether any changes are in progress before escalating.

HOW CAN WE DETERMINE WHAT IS ON A CHANNEL/CHPID?

We have several MVS commands to trace devices. We can trace from the device back to the channel or from the channel down to the device. The approach we will use depends on the type of message we receive and which direction we have to research.

USING MVS COMMANDS TO RESEARCH DEVICES AND CHPIDS

Suppose we get a device error message like:

```
IOS000I 87D4,19,IOE,02,0600,,**,,HSM
```

First, we would use the *Messages and Code* manual (or MVS/Quickref) to determine what the message meant. This particular message will always contain the device number (also known as a UCB). 87D4 is the device number and 19 is the CHPID. Next, you might want to determine what kind of device this is by using the DISPLAY UNIT command:

```
D U,,,87D4,1

IEE457I 07.27.05 UNIT STATUS 420
UNIT TYPE STATUS        VOLSER     VOLSTATE
87D4 359L O    -M                      /REMOV
```

This device is a 359L (logical 3590 in a virtual tape server). We tend to keep the same types of device isolated on a CHPID. If one device is a tape, the others are probably tapes also. Although this is not 100% true, it is a good rule-of-thumb; but always check. The reason this is important is that it gives us a quick feel for what types of device will be affected. Depending on what type of device is on the channel, we may be more or less likely to sustain the hit.

If some other message presents a device number without the CHPID, you could also do a DISPLAY MATRIX command for the device

(also called a DM DEV):

```
D M=DEV(87D4)

IEE174I Ø7.32.28 DISPLAY M 499
DEVICE 87D4   STATUS=ONLINE
CHP                   19
DEST LINK ADDRESS     64
DEST LOGICAL ADDRESS  ØØ
PATH ONLINE           Y
CHP PHYSICALLY ONLINE Y
PATH OPERATIONAL      Y
MANAGED               N
MAXIMUM MANAGED CHPID(S) ALLOWED:  Ø
ND          = ØØ3590.A5Ø.IBM.13.ØØØØØØØ44712
DEVICE NED = ØØ3590.E1A.IBM.13.ØØØØØØØ44712
```

We can see from the third line that this device is on CHPID 19 (with no alternative paths).

Most of our DASD will be configured with multiple CHPIDs for throughput and redundancy:

```
D U,,,A123,1

IEE457I Ø7.35.45 UNIT STATUS 6Ø2
UNIT TYPE STATUS        VOLSER     VOLSTATE
A123 339Ø O             1GA123     PRIV/RSDNT

D M=DEV(A123)

IEE174I Ø7.34.5Ø DISPLAY M 599
DEVICE A123   STATUS=ONLINE
CHP                   A2 D2 62 1F B6
DEST LINK ADDRESS     Ø6 Ø5 Ø4 Ø5 Ø5
DEST LOGICAL ADDRESS  Ø1 Ø1 Ø1 Ø1 Ø1
PATH ONLINE           Y  Y  Y  Y  Y
CHP PHYSICALLY ONLINE Y  Y  Y  Y  Y
PATH OPERATIONAL      Y  Y  Y  Y  Y
MANAGED               N  N  N  N  N
MAXIMUM MANAGED CHPID(S) ALLOWED:  Ø
ND          = ØØ2105.   .HTC.12.ØØØØØØØ40358
DEVICE NED =   2105.   .HTC.12.ØØØØØØØ40358
```

DASD A123 has five paths (CHPIDs A2, D2, 62, 1F, and B6). If one of these CHPIDs has a failure and all the devices on the failing CHPID are configured with the same five CHPIDs, this problem will have minimal impact. There is the potential for a 20% performance

hit, but there should be no loss of functionality. This problem could most likely be deferred until after hours.

Suppose we get a message like this:

```
IOS581E LINK FAILED REPORTING CHPID=A2 INCIDENT UNIT TM=ØØ9Ø32/ØØ5
SER=IBMØ2-Ø41278 IF=ØØØ5 IC=Ø3 INCIDENT UNIT LIF=Ø9
```

This means we have detected a channel/CHPID failure. The quickest way to determine what is on the CHPID is to use the DISPLAY MATRIX command again for the CHPID.

```
D M=CHP(A2)

IEE174I Ø7.42.38 DISPLAY M 65Ø
CHPID A2:  TYPE=Ø5, DESC=ESCON SWITCHED POINT TO POINT, ONLINE
DEVICE STATUS FOR CHANNEL PATH A2
    Ø  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
A1Ø +@ +@ +  +  +  +  +  +  +  +  +  +  +  +  +  +
A11 +  +  +@ +@ +  +  +  +  +  +  +  +  +  +  +  +
A12 +  +  +  +  +  +  +  +  +  +  +  +  +  +  +  +
A13 +  +  +  +  +  +  +  +  +  +  +  +  +  +  +  $@
A14 $@ $@ $@ $@ $@ $@ $@ $@ AL AL AL AL AL AL AL AL
A15 AL AL AL AL AL AL AL AL AL AL AL AL AL AL AL AL
A16 AL AL AL AL AL AL AL AL AL AL AL AL AL AL AL AL
A17 AL AL AL AL UL AL AL UL UL UL AL AL AL AL AL AL
.
. several lines removed from the command output
.
AB8 AL AL AL AL AL AL AL AL AL AL AL AL AL AL AL AL
AB9 AL AL AL AL AL AL AL AL AL AL AL AL AL AL AL AL
ABA AL AL AL AL AL AL AL AL AL AL AL AL AL AL AL AL
ABB AL AL AL AL AL AL AL AL AL AL AL AL AL AL AL AL
ABC AL AL AL AL AL AL AL AL AL AL AL AL AL AL AL AL
ABD AL AL AL AL AL AL AL AL AL AL AL AL AL AL AL AL
ABE AL AL AL AL AL AL AL AL AL AL AL AL AL AL AL AL
ABF AL AL AL AL AL AL AL AL AL AL AL AL AL AL AL AL
SWITCH DEVICE NUMBER = 9Ø12
************************ SYMBOL EXPLANATIONS ************************
+ ONLINE    @ PATH NOT VALIDATED   - OFFLINE    . DOES NOT EXIST
* PHYSICALLY ONLINE   $ PATH NOT OPERATIONAL
BX DEVICE IS BOXED           SN SUBCHANNEL NOT AVAILABLE
DN DEVICE NOT AVAILABLE      PE SUBCHANNEL IN PERMANENT ERROR
AL DEVICE IS AN ALIAS        UL DEVICE IS AN UNBOUND ALIAS
```

USING THE CHPID SPREADSHEET TO RESEARCH CHPIDS

The D M=CHP(xx) shows the larger picture of what is on the

CHPID. This is very complete, but it can be overwhelming, tedious, and time-consuming to research. Most shops maintain a set of spreadsheets to document each CHPID by CEC by data centre. These spreadsheets can help identify the use of a CHPID very quickly. Figure 1 shows what our spreadsheet looks like.

Legend:

#1 Excel tabs for each CEC.

#2 CHPID numbers.

#3 Device number found on that CHPID.

#4 The device types found on that CHPID.

#5 If this is a CF CHPID, the heading will have a blue background.

To find the CHPID in question:

1 Select the tab for the correct CEC.

2 Scroll to the correct CHPID.

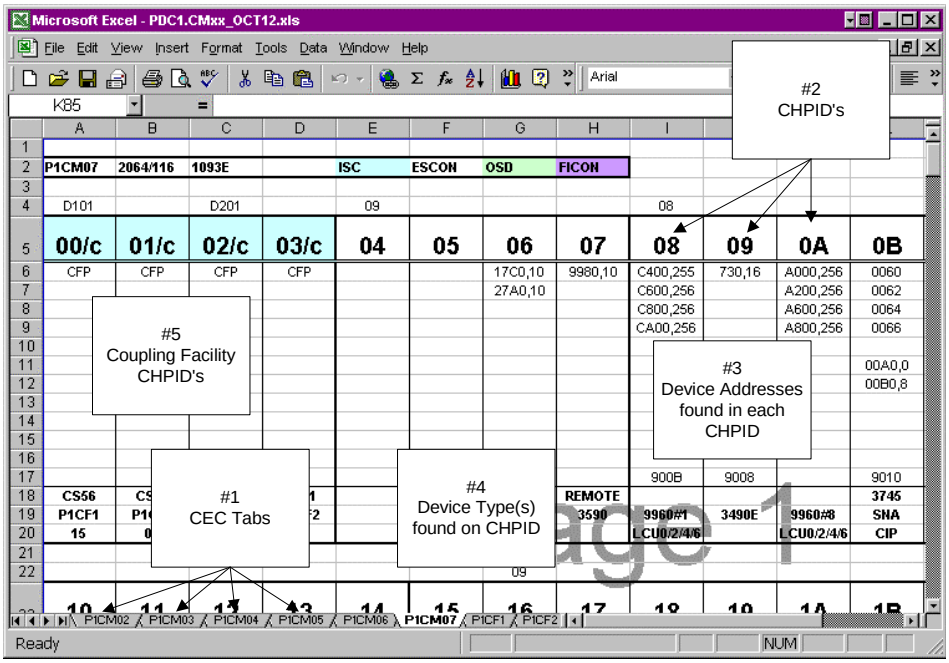

Figure 1: CHPID spreadsheet

3 Review the device types and device addresses for the CHPID.

4 Determine whether known changes are in progress for this CHPID or device range.

5 Assess whether this is a problem that needs immediate attention.

WHAT IF IBM CALLS AND SAYS WE NEED TO REPLACE AN I/O CARD?

Always match serial numbers to CECs to determine the affected LPARs before allowing IBM to do anything. The CHPID is the logical name for a channel. The channel is a fibre cable that is plugged into a port in the CEC. The CEC has 'cages' containing cards with ports. Each CHPID is actually a fibre-channel cable that plugs into an associated port in a card in a cage. A cage is just a frame in the CEC that holds cards. The type of port and the actual location of the port the cable plugs into is based on the IOCDS and the type of card that supports the desired channel/device type. IBM provides different channel cards for the different types of device. For example, cards that support DASD are different from cards that support coupling facilities. Each type of card is also referred to as a Self-Timed Interface (STI). We have ordered all the appropriate STIs for our machines and 'genned' the system to use all those devices.

To keep all this straight, there is a set of Word documents that were provided during the IBM system assurance process when the CECs were installed (CHPID mapping tool). These are in a shared folder and show all the CHPIDs and which ports they plug into. This is important because each cage contains a different mix of cards. Some cards support multiple CHPIDs, so an error on one CHPID does not mean the STI is available for replacement. There must be research to determine whether the STI is shared by multiple CHPIDs. STIs are replaceable while the machine is up and running provided it is possible to VARY/CONFIG all the devices and associated CHPIDs OFFLINE. This may or may not be possible based on the devices on the CHPID. For instance if the paging packs are on a shared STI with a bad port it is not likely that this can be replaced without a maintenance window. If the STI is pulled out while other CHPIDs are active, we will have serious problems.

FINDING AND READING THE CHPID MAPPING TOOL

The CHPID mapping tool allows you to research the location and STI for a CHPID. This should be used if IBM calls to determine whether a concurrent STI replacement can occur.

Figure 2 shows to use the CHPID report:

#1 CHPID from the error message or CHPID being researched.

#2 The slot the cable is plugged into in the cage.

#3 The cage in the mainframe.

#4 The STI that supports the slot.

#5 The IBM part/card number.

How to determine which STI a CHPID is on:

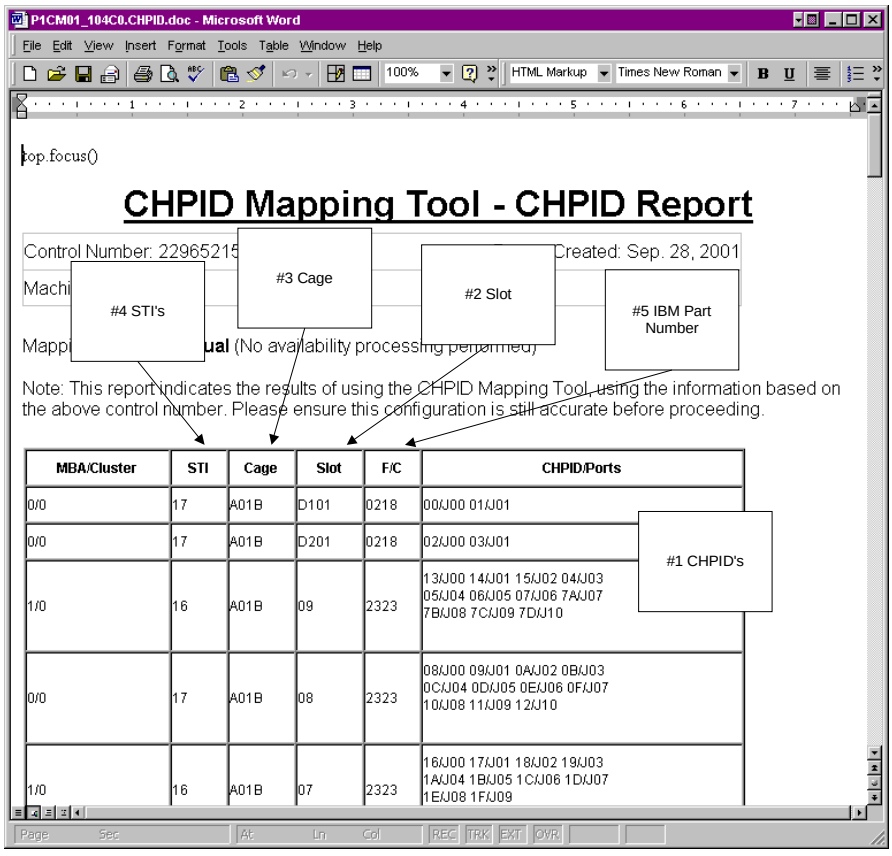

Figure 2: Using the CHPID report

1 Find the CHPID in question

2 Look up the slot, cage, and STI.

How to determine whether the STI is shared:

1 If more than one CHPID is listed for the STI/cage/slot, it is shared.

2 If more than one cage/slot is listed for the STI, it is shared.

Figure 3 shows how to use the CHPID placement report:

#1 The slot identified on the CHPID report.

#2 The port used by each CHPID.

How to determine whether a single port or an entire STI is bad:

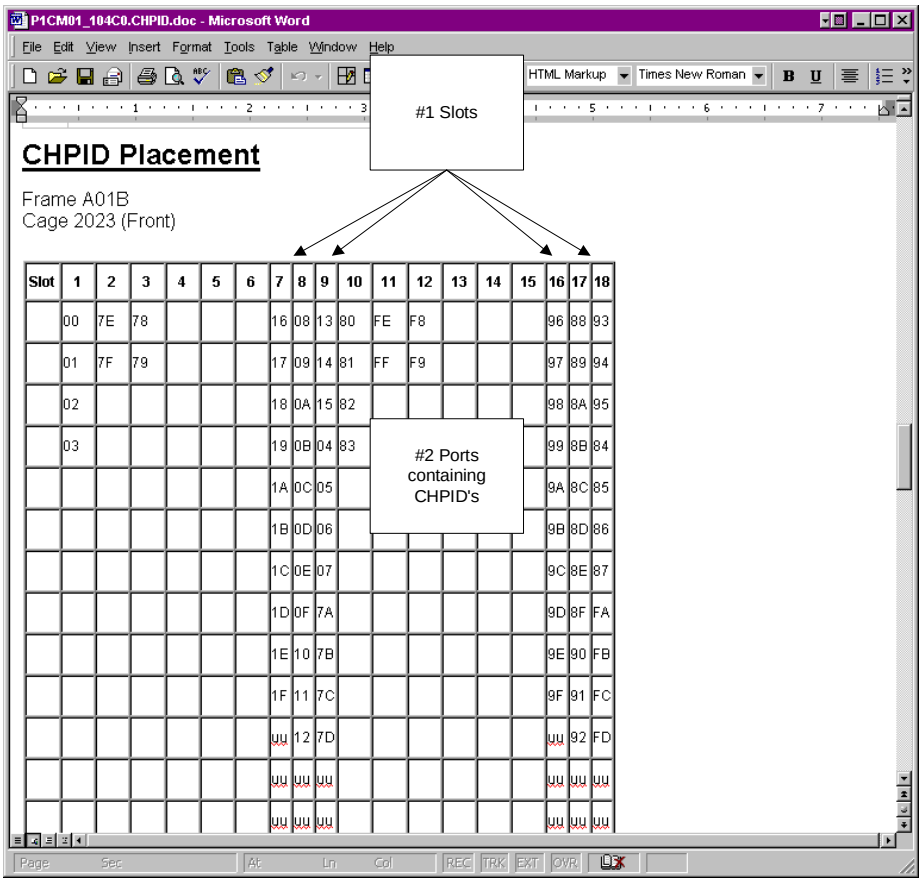

Figure 3: CHPID placement report

1 Take the cage/slot from the CHPID report and scroll forward to the CHPID placement diagram.

2 Look up other CHPIDs on the slot (column).

3 If the others are working, this is a port problem.

How to prepare an STI for replacement without an IPL:

1 Under the direction of OSVS.

2 Research each CHPID on the STI and see if it can be VARY'd OFFLINE.

3 If so, VARY all the CHPIDs devices OFFLINE.

4 CONFIG the CHPID OFFLINE.

5 Repeat for every port/CHPID on the STI.

HOW DOES THIS REALLY WORK?

Here is a complete example of determining whether it is possible to get all CHPIDs/devices on an STI off-line for concurrent maintenance or if a maintenance window is needed.

Example

IBM calls and says serial number 104C0 had a hit on card number 2323 on STI 16 in cage A01B. What do you do?

1 Using the CEC to LPAR to serial number mapping report, find the CEC.

2 Using the CEC to LPAR configuration chart, determine which LPARs will be affected.

3 Find the CHPID mapping tool and open the right document for the CEC.

4 Using the CHPID report, find STI 16.

5 Still using the CHPID report, locate all CHPIDs using STI 16

(remember, an STI can span multiple slots and can contain several CHPIDs).

6 Using the CHPID spreadsheets, locate each CHPID and determine the device type.

7 Issue D M=CHP(xx) commands to determine the device statuses.

8 If DASD or TAPE and ONLINE, determine whether it is realistic to take the devices OFFLINE.

If the DASD can be taken OFFLINE:

1 VARY all the appropriate ranges OFFLINE to all LPARs on the CEC.

2 CONFIG all the CHPIDs OFFLINE to all LPARs on the CEC.

3 Turn over the CEC to IBM.

If the DASD cannot be taken OFFLINE, schedule a maintenance window.

ARE COUPLING FACILITY CHPIDS ANY DIFFERENT?

Yes, a CF CHPID is used exclusively by a coupling facility. The CHPID on the LPAR side is called a sender path and the CHPID on the CF side is called a receiver path. You can see the sender paths from the LPAR only by using the D CF command or a D M=CHP(xx) on a CF CHPID. The only way to see the receiver path is to reference diagrams that show what is connected to what. Usually CF sender path CHPID problems can be fixed by CONFIGing the CHPID off-line and on-line. The CONFIG CHP(xx),ONLINE can take a few minutes to complete. This should be done only if another CF sender path is available and ONLINE to the same CF. Otherwise, this should be done only after a CF is ‘drained’ of all structures and under the supervision of OSVS.

Occasionally, it is necessary to resolve this problem from the CF using CFCC commands to the CF from the HMC. Here is an example of ‘fixing’ a CF CHPID after an IPL.

RESOLVING CF CHPID CONNECTIVITY PROBLEMS

First, attempt to resolve the problem from the LPAR side.

From the LPAR with the CF connectivity problem:

- Confirm the CHPIDs in use by the CF by using the D CF command.
- The CFNAME can be found by finding the NAMED keyword.
- The CFCHPIDs can be found by finding the SENDER keyword.

```
V PATH(CFNAME,CFCHPID),OFFLINE
```

- Wait until the path comes off-line (MVS message IXL101I):

```
CF CHP(CFCHPID),OFFLINE
```

- Wait until the CHPID comes off-line (MVS messages IEE503I and IEE712I), then attempt to bring it back on-line:

```
CF CHP(CFCHPID),ONLINE
```

- It may take a few minutes to complete (goes NOT OPERATIONAL first):

```
V PATH(CFNAME,CFCHPID),ONLINE
```

If the above commands do not fix the problem repeat the sequence and bounce the CF side while the LPAR CHPID is OFFLINE:

```
V PATH(CFNAME,CFCHPID),OFFLINE
CF CHP(CFCHPID),OFFLINE
```

Go to the HMC and bounce the CF RECEIVER PATH on the CF using CFCC commands:

1 Log on to the HMC.

2 Drill into the IPL work area for the correct data centre.

3 Highlight the CF.

4 Double-click on *Operating System Messages* in the *Daily* pane.

5 Click the *Send Command* button.

6 CONFIGURE cfchpid OFFLINE and press *Enter* (use CF CHPID).

7 Wait until it comes off-line (CF message CF0149I).

8 CONFIGURE cfchpid ONLINE and press *Enter*.

9 You may receive an error message (CF0264I Link Failed – CHPID cfchpid).

10 Confirm that the CHPID is on-line with the DISPLAY CHPID ALL command.

11 You should see the CHPID listed in the CF0106I message.

Then return to the LPAR and bring the CF CHPID back ONLINE:

```
CF CHP(CFCHPID),ONLINE
V PATH(CFNAME,CFCHPID),ONLINE
```

Contact hardware support if this does not fix the problem.

ARE CTC CHPIDS ANY DIFFERENT?

Yes, CTCs are owned by VTAM for Channel Adapters (Cross Domain CTCs) and MPC+ channels (more common in APPN CP to CP connections). If, after attempting to resolve connectivity problems through all the normal VTAM commands, the CTCs still will not connect, try using a D M=DEV command and ESCON manager to confirm that everything is mapped correctly in the IOCDS and cabled correctly though the ESCON directors.

If everything is mapped correctly you will see the CECs, CHPIDs, and PORTs matching up in displays from each system. The PORT NAME is made up of the CEC name and the CHPID ID, and the PORTs on each side should point to each other and the TYPEs should be CTC_S on one side and CNC_S on the other.

In this example, LPAR1 (SYS1) on CEC06 has a CTC (0FAE) to LPAR4 (SYS4) on CEC03. The CHPID on the SYS1 side is EF and the CHPID on the SYS4 side is D9. The ESCON director port patched to SYS1 is AC and the port patched to SYS4 is 95.

```
ROUTE SYS1,D M=DEV(ØFAE)
IEE174I 16.56.5Ø DISPLAY M 877
DEVICE ØFAE   STATUS=ONLINE
```

```
CHP                   EF
DEST LINK ADDRESS     95
ENTRY LINK ADDRESS    AC
PATH ONLINE           Y
CHP PHYSICALLY ONLINE Y
PATH OPERATIONAL      Y
DESTINATION CU LOGICAL ADDRESS = Ø4
CU ND      = NOT AVAILABLE
DEVICE NED = ØØ2Ø64.CTC.IBM.Ø2.9542AØØ1Ø93D

ROUTE SYS1,F IHVPROC,D D ØFAE *

IHVC999I ESCON MANAGER DISPLAY 233
IHVC824I                                PORT
IHVC825I          CHP   SWCH           STATUS
IHVC826I DEVN CHP TYPE  DEVN LSN PORT H S C  P PORT NAME
IHVC827I ØFAE EF  CTC_S 9Ø1Ø 17   AC         P CECØ6.CHPEF.CTC/CNC
IHVC82AI CNTL UNIT DATA:9Ø1Ø 17   95         P CECØ3.CHPD9.CTC/CNC
IHVOØØØI I/O-OPS IS READY TO PROCESS OPERATOR COMMANDS

ROUTE SYS4,D M=DEV(ØFAE)
IEE174I 16.57.22 DISPLAY M 154
DEVICE ØFAE   STATUS=ONLINE
CHP                   D9
DEST LINK ADDRESS     AC
ENTRY LINK ADDRESS    95
PATH ONLINE           Y
CHP PHYSICALLY ONLINE Y
PATH OPERATIONAL      Y
DESTINATION CU LOGICAL ADDRESS = Ø2
CU ND      = NOT AVAILABLE
DEVICE NED = ØØ2Ø64.CTC.IBM.Ø2.9542BØØ1Ø93D

ROUTE SYS4,F IHVPROC,D D ØFAE *

IHVC999I ESCON MANAGER DISPLAY 746
IHVC824I                                PORT
IHVC825I          CHP   SWCH           STATUS
IHVC826I DEVN CHP TYPE  DEVN LSN PORT H S C  P PORT NAME
IHVC827I ØFAE D9  CNC_S 9Ø1Ø 17   95         P CECØ3.CHPD9.CTC/CNC
IHVC82AI CNTL UNIT DATA:9Ø1Ø 17   AC         P CECØ6.CHPEF.CTC/CNC
IHVOØØØI I/O-OPS IS READY TO PROCESS OPERATOR COMMANDS
```

If you can't match things up like this, there is a cabling or IOCDS problem.

Robert Zenuk
Systems Programmer (USA)

Disaster recovery procedure

Recently we had to review our disaster recovery procedure. Previously all the back-ups were done with DFDSS on 3490 cartridges. Now that we have 3590 Magstar devices, we can save more DASD volumes on the same cartridge. The goal of this procedure is to check that all our DASD have a back-up, except those used for test data or volumes without data or volumes with page datasets or JES spool.

For this we do an IDCAMS DCOLLECT, which we sort into two files. The first is sorted by volume name and the second by device number. Afterwards we run a REXX procedure that shows us our configuration from the two dcollect reports.

First we produce a report with the DASD volume names and their device number. Then we use the catalog search interface to get all the files having a dataset name mask BKUP.*.G* because we do our full dump DASD on GDGs. The first qualifier is BKUP, the second is the DASD's name, and the third is the generation number.

We print the list of files found using our search criteria, with the cartridge volume name and creation date. We print a report of back-ups that seem to be too old, older than a number of days specified as a parameter – 21 days is the default.

We create IDCAMS define commands to catalog the non-VSAM back-up dataset names, so it's easier to retrieve them on our restore system. Then we do a matching between the DASD volume and the corresponding back-up dataset name – the second qualifier of the dataset must match the volume. We can do a match on the last version or on a previous version.

Now we may get a list of DASD volumes without back-up. We exclude some, based on our standards, such as some starting with TEST** TT**** RV****.

Now we sort our list of back-ups that we've selected and matched. We do this on the cartridge name and file number (file sequence number on cartridge).

We also create some IEBUPDTE statements to add this report later

in a PDS as documentation. We could also create DFDSS dump commands to back-up the volumes without back-up, but this function is described later.

Next we produce JCL to restore the DASD volumes, using 3590 cartridge and file sequence number. We create a member by 3590 cartridge in a PDS, and there is also an alias for each member. This alias is the DASD name of the first dataset name on the 3590 cartridge.

In the case of a DASD volume on two 3590 cartridges, it goes with the first volume and the next DASD goes to a second cartridge member.

We produce a report with the DASD volumes by device number and a report with the gap between device numbers for which no volumes have been found.

Then we produce a JCL with ICKDSF to initialize the DASD volumes as required by our DRP supplier. We also have some volumes that we do not restore, but we need them to do our work.

To restore our production system, we're using a mini OS/390 system. All the JCL produced here is saved in a library on this mini OS/390. This is one volume that we back-up each week.

We have written a small procedure to eject this last back-up out of the library. All this has been done without a tape manager.

After saving our mini-system we do a logical full dump of our master, user, and OAM catalogs on the same cartridge as our mini system.

Our JCL:

```
//JOBDRP00 JOB  ,CLASS=T,MSGCLASS=X,MSGLEVEL=(1,1),
//         NOTIFY=&SYSUID
//STEP000  EXEC PGM=IEFBR14,REGION=2M
//DRPCNTL  DD   DSN=SYS1.DRPOSXX.CNTL,
//         SPACE=(TRK,0),DISP=(MOD,DELETE)
//*
//STEP010  EXEC PGM=IEFBR14,REGION=4M
//DRPCNTL  DD   DSN=SYS1.DRPOSXX.CNTL,DISP=(NEW,CATLG),
//         DSORG=PO,DCB=(RECFM=FB,LRECL=80,BLKSIZE=0),
//         SPACE=(TRK,(15,5,15)),UNIT=3390,VOL=SER=SOSXXX
//*
```

```
//STEPØ2Ø  EXEC PGM=IDCAMS,REGION=4M
//SYSPRINT DD   SYSOUT=*
//OUTDS    DD   DSN=&&DCOLLECT,DISP=(NEW,PASS),
//          DSORG=PS,DCB=(RECFM=VB,LRECL=644,BLKSIZE=Ø),
//          SPACE=(TRK,(15,5),RLSE),UNIT=VIO
//SYSIN    DD   *
  DCOLLECT  OFILE(OUTDS)  VOLUME(*)  NODATAINFO
/*
//STEPØ3Ø  EXEC PGM=SORT,COND=(Ø,LT)
//SYSOUT   DD   SYSOUT=*
//SYSPRINT DD   SYSOUT=*
//SORTIN   DD   DSN=&&DCOLLECT,DISP=(OLD,PASS)
//SORTOUT  DD   DSN=&&DCOLSORT,DISP=(NEW,PASS),
//          SPACE=(TRK,(15,5),RLSE),UNIT=VIO
//SYSIN   DD *
  RECORD TYPE=V,LENGTH=644
  SORT FIELDS=(29,Ø6,CH,A)
  SUM FIELDS=NONE
//*
//STEPØ4Ø  EXEC PGM=SORT,COND=(Ø,LT)
//SYSOUT   DD   SYSOUT=*
//SYSPRINT DD   SYSOUT=*
//SORTIN   DD   DSN=&&DCOLLECT,DISP=(OLD,PASS)
//SORTOUT  DD   DSN=&&DCOLSORD,DISP=(NEW,PASS),
//          SPACE=(TRK,(15,5),RLSE),UNIT=VIO
//SYSIN   DD *
  RECORD TYPE=V,LENGTH=644
  SORT FIELDS=(81,Ø2,CH,A)
  SUM FIELDS=NONE
//*
//STEPØ5Ø  EXEC PGM=IKJEFT1B,DYNAMNBR=2Ø,REGION=6M
//SYSEXEC  DD   DISP=SHR,DSN=your.rexx.exec
//SYSTSPRT DD   SYSOUT=*
//SYSPRINT DD   SYSOUT=*
//DASDVD   DD   SYSOUT=*
//DASDDV   DD   DSN=&&DASDDV,DISP=(,PASS),
//          DCB=(LRECL=8Ø,RECFM=FB,DSORG=PS),
//          UNIT=VIO,SPACE=(TRK,(5,5))
//DASDBK   DD   DSN=&&DASDBK,DISP=(,PASS),
//          DCB=(LRECL=8Ø,RECFM=FB,DSORG=PS),
//          UNIT=VIO,SPACE=(TRK,(5,5))
//DASDDB   DD   DSN=&&DEFNVSAM,DISP=(,PASS),
//          DCB=(LRECL=8Ø,RECFM=FB,DSORG=PS),
//          UNIT=VIO,SPACE=(TRK,(5,5))
//DASDIN   DD   DSN=&&DASDINIT,DISP=(,PASS),
//          DCB=(LRECL=8Ø,RECFM=FB,DSORG=PS),
//          UNIT=VIO,SPACE=(TRK,(5,5))
//DASDFD   DD   SYSOUT=*
//DASDRS   DD   DSN=&&DASDREST,DISP=(,PASS),
//          DCB=(LRECL=8Ø,RECFM=FB,DSORG=PS),
//          UNIT=VIO,SPACE=(TRK,(5,5))
//DCOLIN   DD   DSN=&&DCOLSORT,DISP=(OLD,PASS)
```

```
//DCOLDN   DD   DSN=&&DCOLSORD,DISP=(OLD,PASS)
//SYSTSIN  DD   *
  DRPXVOLØ BKUP.*.G*   7 21 99
//*
//STEPØ6Ø  EXEC PGM=IEBUPDTE,REGION=4M,PARM='MOD'
//SYSUT1   DD   DISP=SHR,DSN=SYS1.DRPOSXX.CNTL
//SYSUT2   DD   DISP=SHR,DSN=SYS1.DRPOSXX.CNTL
//SYSPRINT DD   DUMMY    SYSOUT=*
//SYSIN    DD   DSN=&&DASDREST,DISP=(OLD,PASS)
//*
//STEPØ7Ø  EXEC PGM=IEBUPDTE,REGION=4M,PARM='MOD'
//SYSUT1   DD   DISP=SHR,DSN=SYS1.DRPOSXX.CNTL
//SYSUT2   DD   DISP=SHR,DSN=SYS1.DRPOSXX.CNTL
//SYSPRINT DD   DUMMY    SYSOUT=*
//SYSIN    DD   DSN=&&DASDINIT,DISP=(OLD,PASS)
//*
//STEPØ8Ø  EXEC PGM=IEBUPDTE,REGION=4M,PARM='MOD'
//SYSUT1   DD   DISP=SHR,DSN=SYS1.DRPOSXX.CNTL
//SYSUT2   DD   DISP=SHR,DSN=SYS1.DRPOSXX.CNTL
//SYSPRINT DD   DUMMY    SYSOUT=*
//SYSIN    DD   DSN=&&DEFNVSAM,DISP=(OLD,PASS)
//*
//STEPØ9Ø  EXEC PGM=IEBUPDTE,REGION=4M,PARM='MOD'
//SYSUT1   DD   DISP=SHR,DSN=SYS1.DRPOSXX.CNTL
//SYSUT2   DD   DISP=SHR,DSN=SYS1.DRPOSXX.CNTL
//SYSPRINT DD   DUMMY    SYSOUT=*
//SYSIN    DD   DSN=&&DASDDV,DISP=(OLD,PASS)
//*
//STEP1ØØ  EXEC PGM=IEBUPDTE,REGION=4M,PARM='MOD'
//SYSUT1   DD   DISP=SHR,DSN=SYS1.DRPOSXX.CNTL
//SYSUT2   DD   DISP=SHR,DSN=SYS1.DRPOSXX.CNTL
//SYSPRINT DD   DUMMY    SYSOUT=*
//SYSIN    DD   DSN=&&DASDBK,DISP=(OLD,PASS)
//*
//STEP2ØØ  EXEC PGM=IKJEFT1B,DYNAMNBR=2Ø,REGION=6M
//SYSEXEC  DD   DISP=SHR,DSN=your.rexx.exec
//SYSTSPRT DD   SYSOUT=*
//DUMPCAT  DD   SYSOUT=*
//DUMPCAT  DD   DSN=&&DUMPCT,DISP=(,PASS),
//         DCB=(LRECL=8Ø,RECFM=FB,DSORG=PS),
//         UNIT=VIO,SPACE=(TRK,(5,5))
//SYSTSIN  DD   *
  DRPXLCAT
//*
//STEP3ØØ  EXEC PGM=ADRDSSU,REGION=6M
//SYSPRINT DD   SYSOUT=*
//DASD     DD   UNIT=339Ø,VOL=SER=SOSXXX,DISP=SHR
//TAPE     DD   DSN=BKUP.SOSXXX(+1),DISP=(,CATLG,DELETE),
//         UNIT=MAG,VOL=(,,,3Ø),
//         DCB=(DSCB,LRECL=32756,BLKSIZE=3276Ø,RECFM=VB,TRTCH=COMP),
//         LABEL=EXPDT=99ØØØ
//SYSIN    DD   *
```

```
  DUMP  FULL  INDDNAME(DASD)  OUTDDNAME(TAPE)  ADMIN CANCELERROR
//*
//STEP31Ø  EXEC PGM=ADRDSSU,REGION=6M
//SYSPRINT DD   SYSOUT=*
//TAPE     DD DSN=BKUP.CAT$$$(+1),DISP=(,CATLG,DELETE),
//          UNIT=(MAG,,DEFER),VOL=(,RETAIN,,99,REF=*.STEP3ØØ.TAPE),
//          DCB=(DSCB,LRECL=32756,BLKSIZE=3276Ø,RECFM=VB,TRTCH=COMP),
//          LABEL=(2,SL),EXPDT=99ØØØ
//SYSIN    DD   DSN=&&DUMPCT,DISP=(OLD,PASS)
//*
//STEP39Ø  EXEC PGM=IKJEFT1B,DYNAMNBR=2Ø,REGION=6M
//SYSEXEC  DD   DISP=SHR,DSN=your.rexx.exec
//SYSTSPRT DD   SYSOUT=*
//SYSTSIN  DD   *
  DRPXEJEØ BKUP.SOSXXX.G*
//
```

REXX PROC to create our restore JCL:

```
/* REXX                                                              */
/*-------------------------------------------------------------------*/
/* Proc drpxvolØ                                                     */
/*   Input : BKPA   -> generic mask for backup datasets              */
/*           NBRD   -> number of days for last backup          7     */
/*           NBRL   -> number of days for oldest backup       21     */
/*           BLM    -> backup limit number Ø1 oldest 99 last  99     */
/*   Output : report with Volume name & device number.               */
/*            report with backup informations.                       */
/*            report with dasd volume without backup.                */
/*-------------------------------------------------------------------*/
 ARG BKPA NBRD NBRL BLM
 TRACE o;
 CALL PROC_PARM;
 CALL PROC_DCOL;
 CALL PROC_NONVSFL;
 CALL PROC_PRTBKPL;
 CALL PROC_DFNVBKP;
 CALL PROC_CHECKBKP;
 CALL PROC_SORTBKP ;
 CALL PROC_DUMPDASD;
 CALL PROC_RESTDASD;
 CALL PROC_SORTDVNO;
 CALL PROC_INITDASD;
 "EXECIO Ø DISKW DASDBK (FINIS";
 "EXECIO Ø DISKW DASDDV (FINIS";
 "EXECIO Ø DISKW DASDVD (FINIS";
 "EXECIO Ø DISKW DASDIN (FINIS";
RETURN;
 /* Proc parm                                              */
PROC_parm:
 IF BKPA = "" THEN HQNVS = "BKUP.*.G*";
              ELSE HQNVS = BKPA;
```

```
 IF NBRD = "" THEN NBRD = 7;
 IF NBRL = "" THEN NBRL = 21;
 IF BLM  = "" THEN BLM  = 99;
 DATE_WKJ = DATE('J');
 DATE_WKS = DATE('S');
 YEAR_BKUP = SUBSTR(DATE_WKS,1,4);
 DAY_BKUP = SUBSTR(DATE_WKJ,3,3) - NBRD;
 IF DAY_BKUP < Ø
 THEN DO;
   DAY_BKUP = 365 - DAY_BKUP;
   YEAR_BKUP = YEAR_BKUP - 1;
 END;
 DAY_BKUP = RIGHT(DAY_BKUP,3,"Ø");
 BKUPD = YEAR_BKUP || DAY_BKUP;
 YEAR_BKUP = SUBSTR(DATE_WKS,1,4);
 DLM_BKUP = SUBSTR(DATE_WKJ,3,3) - NBRL;
 IF DLM_BKUP < Ø
 THEN DO;
   DLM_BKUP = 365 - DLM_BKUP;
   YEAR_BKUP = YEAR_BKUP - 1;
 END;
 Dlm_BKUP = RIGHT(Dlm_BKUP,3,"Ø");
 BKUPl = YEAR_BKUP || Dlm_BKUP;
 say " ******************************************** ";
 SAY "   nbrd : " nbrd ;
 SAY "   date : " date_wks;
 SAY "   date : " date_wkj;
 SAY "   MinD : " DAY_BKUP;
 SAY "   MaxD : " DLM_BKUP;
 SAY "   last : " BKUPD;
 SAY "   old  : " BKUPl;
 SAY "   BkVer: " BLM;
 say " ******************************************** ";
RETURN;
 /* Proc read Dcollect print report volume device number */
PROC_DCOL:
 "EXECIO * DISKR DCOLIN (STEM DCOLV.";
 VI = Ø;
 DO WHILE VI < DCOLV.Ø;
   VI = VI + 1;
   TV.VI = SUBSTR(DCOLV.VI,25,6);
   TD.VI = C2X(SUBSTR(DCOLV.VI,77,2));
   TS.VI = SUBSTR(DCOLV.VI,83,8);
   tb.vi = "?";
 END;
 K = 1;
 R.1 = "  " COPIES("*",71);
 R.2 = "   ** DASD VOLUME WITH DEVICE NUMBER" COPIES(" ",34) "**";
 R.3 = "  " COPIES("*",71);
 "EXECIO 3 DISKW DASDVD (STEM R.";
 DO WHILE K <= VI;
   DASD_O = "     ";
```

```
   DO J = 1 TO 5 WHILE K <= VI;
     DASD_O = DASD_O || TV.K || " " || TD.K || "   ";
     K = K + 1;
   END;
   RECO.1 = DASD_O ;
   "EXECIO 1 DISKW DASDVD (STEM RECO.";
 END;
 R.1 = "  " COPIES("*",71);
 "EXECIO 1 DISKW DASDVD (STEM R.";
RETURN;
 /* Proc NonVS_fl                                        */
PROC_NONVSFL:
 KEY = HQNVS || '.**';
 COUNT = Ø                          /* TOTAL ENTIRES FOUND          */
 MODRSNRC = SUBSTR(' ',1,4)         /*   CLEAR MODULE/RETURN/REASON */
 CSIFILTK = SUBSTR(KEY,1,44)        /*   MOVE FILTER KEY INTO LIST  */
 CSICATNM = SUBSTR(' ',1,44)        /*   CLEAR CATALOG NAME         */
 CSIRESNM = SUBSTR(' ',1,44)        /*   CLEAR RESUME NAME          */
 CSIDTYPS = SUBSTR('ABH',1,16)      /*   CLEAR ENTRY TYPES          */
 CSICLDI  = SUBSTR('Y',1,1)         /*   INDICATE DATA AND INDEX    */
 CSIRESUM = SUBSTR(' ',1,1)         /*   CLEAR RESUME FLAG          */
 CSIS1CAT = SUBSTR(' ',1,1)         /* SEARCH > 1 CATALOGS          */
 CSIRESRV = SUBSTR(' ',1,1)         /*   CLEAR RESERVE CHARACTER    */
 CSINUMEN = 'ØØØ5'X                 /*   INIT NUMBER OF FIELDS      */
 CSIFLD1    = SUBSTR('VOLSER',1,8)  /*   INIT FIELD 1 FOR VOLSERS   */
 CSIFLD2    = SUBSTR('DEVTYP',1,8)  /*   INIT FIELD 2 FOR DEVTYP    */
 CSIFLD3    = SUBSTR('FILESEQ',1,8) /*   INIT FIELD 5 FOR DS EX DT  */
 CSIFLD4    = SUBSTR('DSCRDT2',1,8) /*   INIT FIELD 3 FOR DS CR DT  */
 CSIFLD5    = SUBSTR('DSEXDT2',1,8) /*   INIT FIELD 4 FOR DS EX DT  */
  /*  BUILD THE SELECTION CRITERIA FIELDS PART OF PARAMETER LIST    */
 CSIOPTS  = CSICLDI || CSIRESUM || CSIS1CAT || CSIRESRV
 CSIFIELD = CSIFILTK || CSICATNM || CSIRESNM || CSIDTYPS || CSIOPTS
 CSIFIELD = CSIFIELD || CSINUMEN || CSIFLD1 || CSIFLD2 || CSIFLD3
 CSIFIELD = CSIFIELD || CSIFLD4 || CSIFLD5;
  /*  INITIALIZE AND BUILD WORK ARE OUTPUT PART OF PARAMETER LIST    */
 WORKLEN = 131Ø72  /* 128K */
 DWORK = 'ØØØ2ØØØØ'X || COPIES('ØØ'X,WORKLEN-4)
  /*  INITIALIZE WORK VARIABLES                                     */
 RESUME = 'Y'
 CATNAMET = SUBSTR(' ',1,44)
 DNAMET = SUBSTR(' ',1,44)
 IC = Ø;
  /*  SET UP LOOP FOR RESUME (IF A RESUME IS NCESSARY)              */
 DO WHILE RESUME = 'Y'
  /*  ISSUE LINK TO CATALOG GENERIC FILTER INTERFACE                */
   ADDRESS LINKPGM 'IGGCSIØØ  MODRSNRC  CSIFIELD  DWORK'
   RESUME = SUBSTR(CSIFIELD,15Ø,1);
   USEDLEN = C2D(SUBSTR(DWORK,9,4));
   POS1=15;
  /*  PROCESS DATA RETURNED IN WORK AREA                            */
   DO WHILE POS1 < USEDLEN
     IF SUBSTR(DWORK,POS1+1,1) = 'Ø'
```

```
   THEN DO
     CATNAME=SUBSTR(DWORK,POS1+2,44)
     POS1 = POS1 + 5Ø
   END
   IF POS1 < USEDLEN    /* IF STILL MORE DATA       */
   then DO              /* CONTINUE WITH NEXT ENTRY */
     DNAME = SUBSTR(DWORK,POS1+2,44)  /* GET ENTRY NAME   */
/*  ASSIGN ENTRY TYPE NAME                                          */
     SELECT;
       WHEN SUBSTR(DWORK,POS1+1,1) = 'C' THEN DTYPE = 'CLUSTER '
       WHEN SUBSTR(DWORK,POS1+1,1) = 'D' THEN DTYPE = 'DATA    '
       WHEN SUBSTR(DWORK,POS1+1,1) = 'I' THEN DTYPE = 'INDEX   '
       WHEN SUBSTR(DWORK,POS1+1,1) = 'A' THEN DTYPE = 'NONVSAM '
       WHEN SUBSTR(DWORK,POS1+1,1) = 'H' THEN DTYPE = 'GDS     '
       WHEN SUBSTR(DWORK,POS1+1,1) = 'B' THEN DTYPE = 'GDG     '
       WHEN SUBSTR(DWORK,POS1+1,1) = 'R' THEN DTYPE = 'PATH    '
       WHEN SUBSTR(DWORK,POS1+1,1) = 'G' THEN DTYPE = 'AIX     '
       WHEN SUBSTR(DWORK,POS1+1,1) = 'X' THEN DTYPE = 'ALIAS   '
       WHEN SUBSTR(DWORK,POS1+1,1) = 'U' THEN DTYPE = 'UCAT    '
       OTHERWISE DO;
         IF SUBSTR(DWORK,POS1+1,1) = 'Ø' THEN ITERATE
         POS1 = POS1 + 46
         POS1 = POS1 + C2D(SUBSTR(DWORK,POS1,2))
         ITERATE
       END;
     end;
/* CHECK FOR ERROR IN ENTRY RETURNED                            */
     CSIEFLAG = SUBSTR(DWORK,POS1+Ø,1)
     IF BITAND(CSIEFLAG,'4Ø'X) = '4Ø'X
     then do
       POS1 = POS1 + 5Ø           /* HEADER LENGTH */
       POS1 = POS1 + C2D(SUBSTR(DWORK,POS1,2))
       ITERATE    /* GO TO NEXT ENTRY */
     END
/*  HAVE NAME AND TYPE, GET VOLSER INFO                             */
     COUNT = COUNT + 1  /* total entries found  */
     POS1 = POS1 + 46
     IF DTYPE = 'NONVSAM' | DTYPE = 'GDS'
     THEN DO;
       NUMVOL = C2D(SUBSTR(DWORK,POS1+4,2))/6
       DATAPV = 14;                /* POINTER TO FIRST VOLSER  */
       DATAPD = 14 + (NUMVOL * 6);  /* POINTER TO FIRST DEVTYPE */
       DATAPF = 14 + (NUMVOL * 1Ø); /* POINTER TO FIRST FILESEQ */
       DATAPX = 14 + (NUMVOL * 12); /* POINTER TO OTHER DATA    */
                              /* 12 = 6 VOL + 4 DATE + 2 FILENO */
       POS2 = POS1 + 4 + (NUMVOL * 12) + 1Ø;
       POSD = POS2 + (NUMVOL * 6) + 1Ø;
       POSF = POSD + (NUMVOL * 4) + 1Ø;
       VOL2 = copies('*',6);
       VOL3 = copies('*',6);
       DVT2 = copies('*',8);
       DVT3 = copies('*',8);
```

```
         FSN2 = '***';
         FSN3 = '***';
       /* SAY "DEBUGØØ:"SUBSTR(DWORK,POS1,8Ø); */
         VOL1 = left(substR(DWORK,POS1+DATAPV,6),6," ");
         DVT1 = C2X(SUBSTR(DWORK,POS1+DATAPD,4));
         FSN1 = RIGHT(C2D(SUBSTR(DWORK,POS1+DATAPF,2)),3,"Ø");
         IF NUMVOL > 1
         THEN DO;
           VOL2 = SUBSTR(DWORK,POS1+DATAPV+6,6);
           DVT2 = C2X(SUBSTR(DWORK,POS1+DATAPD+4,4));
           FSN2 = RIGHT(C2D(SUBSTR(DWORK,POS1+DATAPF+2,2)),3,"Ø");
         END;
         IF NUMVOL > 2
         THEN DO;
           VOL3 = SUBSTR(DWORK,POS1+DATAPV+12,6);
           DVT3 = C2X(SUBSTR(DWORK,POS1+DATAPD+8,4));
           FSN3 = RIGHT(C2D(SUBSTR(DWORK,POS1+DATAPF+4,2)),3,"Ø");
         END;
         CRDT = C2X(SUBSTR(DWORK,POS1+DATAPX,4));
         CDDD = SUBSTR(CRDT,3,3);
         CYY = SUBSTR(CRDT,1,2);
         CCC = SUBSTR(CRDT,7,2);
         IF CCC = 'ØØ' THEN CCC = '19';
                      ELSE CCC = '2Ø';
         CRDT = CCC || CYY || CDDD;
         IC = IC + 1;
         RECI.IC = DNAME NUMVOL CRDT, /* EXDT, */
                   VOL1 VOL2 VOL3,
                   FSN1 FSN2 FSN3,
                   DVT1 DVT2 DVT3;
       END;
  /*   GET POSITION OF NEXT ENTRY                                    */
       POS1 = POS1 + C2D(SUBSTR(DWORK,POS1,2))
     END;
   END;
 END;
RETURN;
 /* PROC PRINT BKUP                                        */
PROC_PRTBKPL:
 R.1 = "./         ADD  NAME=DOCBKUP                              ";
 r.2 = "  " copies("*",71);
 r.3 = "   ** Backup extracted from ICF Catalog" copies(" ",31) "**";
 r.4 = "  " copies("*",71);
 r.5 = "   ** Dataset name        HQ1  Volume   GDG  " ||,
       "  # date_Cr Volume Fsn";
 r.6 = "  " copies("*",71);
 "EXECIO 6 DISKW DASDBK (STEM R.";
 io = Ø;
 ib = Ø;
 DO I = 1 TO IC;
   BKP_NM = WORD(RECI.I,1);
   bkp_wk = translate(bkp_nm," ",".");
```

```
   bkp_n1 = word(bkp_wk,1);
   bkp_n2 = word(bkp_wk,2);
   bkp_n3 = word(bkp_wk,3);
   bkp_nv = word(reci.i,2);
   bkp_cr = word(reci.i,3);
   bkp_v1 = word(reci.i,4);
   bkp_v2 = word(reci.i,5);
   bkp_v3 = word(reci.i,6);
   bkp_f1 = word(reci.i,7);
   bkp_f2 = word(reci.i,8);
   bkp_f3 = word(reci.i,9);
   bkp_dt = word(reci.i,10);
   if bkp_cr >= bkupl
   then do;
     r.1 = "   *" bkp_nm bkp_n1 bkp_n2 bkp_n3,
          bkp_nv bkp_cr bkp_v1 bkp_f1;
     if substr(bkp_n2,1,1) ¬= "?"
     then "EXECIO 1 DISKW DASDBK (STEM R.";
     IB = IB + 1;
     TB_BKP_NM.IB = BKP_NM;
     TB_BKP_CR.IB = BKP_CR;
     TB_BKP_N2.IB = BKP_N2;
     TB_BKP_V1.IB = BKP_V1;
     TB_BKP_F1.IB = BKP_F1;
   END;
   ELSE DO;
     IO = IO + 1;
     TBO.IO = "   *" BKP_NM BKP_CR;
   END;
 END;
 if io > 0
 then do;
   R.1 = "  " COPIES("*",71);
   R.2 = "   ** BACKUP TOO OLD " COPIES(" ",49) "**";
   R.3 = "  " COPIES("*",71);
   R.4 = "   ** DATASET NAME        DATE_CR";
   R.5 = "  " COPIES("*",71);
   "EXECIO 5 DISKW DASDBK (STEM R.";
   "EXECIO" IO " DISKW DASDBK (STEM TBO.";
 END;
RETURN;
 /* PROC define non-vsam for bkup                          */
PROC_DFNVBKP:
 DEF.1 = "./         ADD  NAME=DEFNVSAM                                ";
 "EXECIO 1 DISKW DASDDB (STEM DEF.";
 DO I = 1 TO IC;
   BKP_NM = WORD(RECI.I,1);
   BKP_WK = TRANSLATE(BKP_NM," ",".");
   BKP_N1 = WORD(BKP_WK,1);
   BKP_N2 = WORD(BKP_WK,2);
   BKP_N3 = WORD(BKP_WK,3);
   BKP_NV = WORD(RECI.I,2);
```

```
   BKP_CR = WORD(RECI.I,3);
   BKP_V1 = WORD(RECI.I,4);
   BKP_V2 = WORD(RECI.I,5);
   BKP_V3 = WORD(RECI.I,6);
   BKP_F1 = WORD(RECI.I,7);
   BKP_F2 = WORD(RECI.I,8);
   BKP_F3 = WORD(RECI.I,9);
   DEV = WORD(RECI.I,1Ø);
   IF BKP_CR >= BKUPL
   THEN DO;
     BKP_WK = TRANSLATE(BKP_NM," ",".");
     BKP_GDG = WORD(BKP_WK,1);
     DO K = 2 TO WORDS(BKP_WK) - 1;
       BKP_GDG = BKP_GDG || "." || WORD(BKP_WK,K);
     END;
     DEF.1 = "  DEFINE GENERATIONDATAGROUP                  -";
     DEF.2 = "        (NAME(" || BKP_GDG || ")              -";
     DEF.3 = "        LIMIT(3) NOEMPTY SCRATCH )            -";
     DEF.4 = "        CATALOG('CATALOG.MVSICF1.VESAØØ2') ";
     VOLW = BKP_V1;
     IF SUBSTR(BKP_V2,1,1) ¬= '*'
     THEN VOLW = VOLW || " " || BKP_V2;
     IF SUBSTR(BKP_V3,1,1) ¬= '*'
     THEN VOLW = VOLW || " " || BKP_V2;
     FEQW = BKP_F1;
     IF SUBSTR(BKP_F2,1,1) ¬= '*'
     THEN FEQW = FEQW || " " || BKP_F2;
     IF SUBSTR(BKP_F3,1,1) ¬= '*'
     THEN FEQW = FEQW || " " || BKP_F3;
     DEF.5 =  "  DEFINE NONVSAM (NAME("BKP_NM") -";
     IF DEV = "78Ø48Ø83"
     THEN DEVT = "DEVT(359Ø-1)";
     ELSE DEVT = "DEVT(349Ø)";
     DEF.6 =  "          " || DEVT || " VOLUMES("VOLW") FSEQN("FEQW"))";
     "EXECIO 6 DISKW DASDDB (STEM DEF.";
   END;
 END;
 "EXECIO Ø DISKW DASDDB (FINIS";
RETURN;
 /* PROC check bkup & volume                                */
PROC_CHECKBKP:
 dnb = Ø;
 bnd = Ø;
 KV = 1;
 KC = 1;
 DO WHILE KV < VI & KC < IC;
   BKP_NM = WORD(RECI.KC,1);
   BKP_WK = TRANSLATE(BKP_NM," ",".");
   BKP_N1 = WORD(BKP_WK,1);
   BKP_N2 = WORD(BKP_WK,2);
   BKP_N3 = WORD(BKP_WK,3);
   IF TV.KV = BKP_N2 & BKP_CR >= BKUPD
```

```
  THEN DO;
    KCL = KC;
    BKP_C2 = BKP_N2;
    BI = 1;
    DO WHILE KCL < IC & BKP_C2 = BKP_N2 & BI <= BLM;
      KCL = KCL + 1;
      BKP_NM = WORD(RECI.KCL,1);
      BKP_WK = TRANSLATE(BKP_NM," ",".");
      BKP_N2 = WORD(BKP_WK,2);
      BI = BI + 1;
    END;
    KC = KCL - 1;
    TB.KV = RECI.KC;
    KV = KV + 1;
    KC = KC + 1;
  END;
  ELSE DO;
    IF TV.KV < SUBSTR(bkp_nm,6,6)
    THEN DO;
      dnb = dnb + 1;
      tnb.dnb = tv.kv;
      tnn.dnb = td.kv;
      TB.KV = "?";
      KV = KV + 1;
    END;
    else DO;
      bnd = bnd + 1;
      tnd.bnd = filenm;
      KC = KC + 1;
    END;
  END;
END;
DO WHILE KV < VI;
  dnb = dnb + 1;
  tnb.dnb = tv.kv;
  tnn.dnb = td.kv;
  TB.KV = "?";
  KV = KV + 1;
END;
DO WHILE KC < IC;
  FILENM = WORD(RECI.KC,1);
  bnd = bnd + 1;
  tnd.bnd = bkp_nm;
  KC = KC + 1;
END;
K = 1;
r.1 = "  " copies("*",71);
r.2 = "   ** Dasd volume without backup exc(TESTxx TTxxxx RVxxxx   *";
r.3 = "  " copies("*",71);
"EXECIO 3 DISKW DASDBK (STEM R.";
DO WHILE K <= dnb;
  DASD_O = "     ";
```

```
   DO J = 1 TO 5 WHILE K <= dnb;
     if substr(tnb.k,1,4) = "TEST" |,
        substr(tnb.k,1,6) = "RV" || tnn.k |,
        substr(tnb.k,1,6) = "TT" || tnn.k
     then j = j - 1;
     else DASD_O = DASD_O || TNB.K || " " || TNN.K || "   ";
     K = K + 1;
   END;
   r.1 = DASD_O;
   "EXECIO 1 DISKW DASDBK (STEM R.";
 end;
 r.1 = "  " copies("*",71);
 "EXECIO 1 DISKW DASDBK (STEM R.";
RETURN;
 /* PROC build init volume                                  */
PROC_SORTDVNO:
 "EXECIO * DISKR DCOLDN (STEM DCOLV.";
 VI = Ø;
 DO WHILE VI < DCOLV.Ø;
   VI = VI + 1;
   TV.VI = SUBSTR(DCOLV.VI,25,6);
   TD.VI = C2X(SUBSTR(DCOLV.VI,77,2));
   DD.VI = X2D(TD.VI);
   TS.VI = SUBSTR(DCOLV.VI,83,8);
 END;
 K = 1;
 r.1 = "./         ADD  NAME=DOCDASD                              ";
 r.2 = "  " copies("*",71);
 r.3 = "   ** Dasd volume with device number" copies(" ",34) "**";
 r.4 = "  " copies("*",71);
 "EXECIO 4 DISKW DASDDV (STEM R.";
 DO WHILE K <= VI;
   DASD_O = "     ";
   DO J = 1 TO 4 WHILE K <= VI;
     DASD_O = DASD_O || TV.K || " " || TD.K || "   ";
     K = K + 1;
   END;
   RECO.1 = DASD_O ;
   "EXECIO 1 DISKW DASDDV (STEM RECO.";
 end;
 R.1 = "  " COPIES("*",71);
 R.2 = "   ** MISSING DEVICE NUMBER BETWEEN" COPIES(" ",32) "**";
 R.3 = "  " COPIES("*",71);
 "EXECIO 3 DISKW DASDDV (STEM R.";
 PD = DD.1;
 DO K = 2 TO VI;
   I = K - 1;
   IF PD+1 < DD.K
   then do;
     r.1 = "      " TV.I TD.I " - " TV.K TD.K ;
     "EXECIO 1 DISKW DASDDV (STEM R.";
   END;
```

```
   PD = DD.K;
 END;
 r.1 = "  " COPIES("*",71);
 r.2 = "  " COPIES("*",71);
 r.3 = "   ** DEVICE NUMBER & VOLUME ERROR " COPIES(" ",32) "**";
 r.4 = "  " COPIES("*",71);
 "EXECIO 4 DISKW DASDDV (STEM R.";
 DO K = 1 TO VI;
   IF SUBSTR(TD.K,1,1) = '4'
   THEN DO;
     IF SUBSTR(TV.K,1,4) = 'TEST' |,
        SUBSTR(TV.K,1,4) = 'DB2D' |,
        SUBSTR(TV.K,1,4) = 'DBCS' |,
        TV.K = "TT" || TD.K
     THEN NOP;
     else do;
       r.1 = "      " TD.K TV.K ;
       "EXECIO 1 DISKW DASDDV (STEM R.";
     END;
   END;
 END;
 r.1 = "  " COPIES("*",71);
 "EXECIO 1 DISKW DASDDV (STEM R.";
RETURN;
 /* PROC sort  bkup & volume                              */
PROC_SORTBKP:
 DO K = 1 TO VI;
   TB_BKP_NM.K = WORD(TB.K,1);
   BKP_NM = WORD(TB.K,1);
   BKP_WK = TRANSLATE(BKP_NM," ",".");
   BKP_N2 = WORD(BKP_WK,2);
   TB_BKP_N2.K = BKP_N2;
   TB_BKP_NV.K = WORD(TB.K,2);
   TB_BKP_CR.K = WORD(TB.K,3);
   TB_BKP_V1.K = WORD(TB.K,4);
   TB_BKP_V2.K = WORD(TB.K,5);
   TB_BKP_V3.K = WORD(TB.K,6);
   TB_BKP_F1.K = WORD(TB.K,7);
   TB_BKP_F2.K = WORD(TB.K,8);
   TB_BKP_F3.K = WORD(TB.K,9);
 /*IF SUBSTR(TB_BKP_NM.K,1,1) = "?" THEN K = K - 1;*/
 END;
 IB = VI;
 I = IB - 1;
 DO WHILE I > Ø;
   J = 1;
   DO WHILE J < I;
     K = J + 1;
     IF (TB_BKP_V1.J > TB_BKP_V1.K) |,
        (TB_BKP_V1.J = TB_BKP_V1.K & TB_BKP_F1.J > TB_BKP_F1.K)
     THEN DO;
       TM_BKP_NM = TB_BKP_NM.J;
```

```
        TM_BKP_CR = TB_BKP_CR.J;
        TM_BKP_N2 = TB_BKP_N2.J;
        TM_BKP_V1 = TB_BKP_V1.J;
        TM_BKP_F1 = TB_BKP_F1.J;
        TM_BKP_V2 = TB_BKP_V2.J;
        TM_BKP_F2 = TB_BKP_F2.J;
        TB_BKP_NM.J = TB_BKP_NM.K;
        TB_BKP_CR.J = TB_BKP_CR.K;
        TB_BKP_N2.J = TB_BKP_N2.K;
        TB_BKP_V1.J = TB_BKP_V1.K;
        TB_BKP_F1.J = TB_BKP_F1.K;
        TB_BKP_V2.J = TB_BKP_V2.K;
        TB_BKP_F2.J = TB_BKP_F2.K;
        TB_BKP_NM.K = TM_BKP_NM;
        TB_BKP_CR.K = TM_BKP_CR;
        TB_BKP_N2.K = TM_BKP_N2;
        TB_BKP_V1.K = TM_BKP_V1;
        TB_BKP_F1.K = TM_BKP_F1;
        TB_BKP_V2.K = TM_BKP_V2;
        TB_BKP_F2.K = TM_BKP_F2;
      END;
      J = J + 1;
    END;
    I = I - 1;
  END;
  R.1 = "./         ADD  NAME=DOCBKUPV                              ";
  r.2 = "  " copies("*",71);
  r.3 = "   ** All backup to restore by volume" copies(" ",33) "**";
  r.4 = "  " copies("*",71);
  "EXECIO 4 DISKW DASDBK (STEM R.";
  DO K = 1 TO IB;
  R.1="   *" TB_BKP_V1.K TB_BKP_F1.K TB_BKP_NM.K TB_BKP_N2.K TB_BKP_V2.K;
    IF TB_BKP_V1.K ¬= " "
    THEN "EXECIO 1 DISKW DASDBK (STEM R.";
  END;
  r.1 = "  " copies("*",71);
  "EXECIO 1 DISKW DASDbk (STEM R.";
RETURN;
  /* PROC build init volume execpt SOSSxx                 */
PROC_INITDASD:
  JX.1 = "./         ADD  NAME=INITDASD                              ";
  JO.1 = "//INITDASD JOB  ,'INITDASD',MSGLEVEL=(1,1),                ";
  JO.2 = "//         MSGCLASS=X,CLASS=S,TYPRUN=SCAN                  ";
  JO.3 = "//* INIT DASD DISASTER RECOVERY                            ";
  SP.1 = "//INIT????  EXEC PGM=ICKDSF,REGION=4M                      ";
  SP.2 = "//SYSPRINT  DD   SYSOUT=*                                  ";
  SP.3 = "//SYSIN     DD   *                                         ";
  SP.4 = "  INIT UNIT(????) MAP NOCHECK NORECLAIM          -         ";
  SP.5 = "       NOVERIFY PURGE VOLID(??????) ???????????? -         ";
  SP.6 = "       NOBOOTSTRAP VTOC(1,3,72) INDEX(Ø,1,14)              ";
  SP.7 = "/*                                                         ";
  "EXECIO 1 DISKW DASDIN (STEM JX.";
```

```
 pxxx = "";
 DO K = 1 TO VI;
   IF PXXX ¬= SUBSTR(TD.K,1,3)
   THEN DO;
     "EXECIO 3 DISKW DASDIN (STEM JO.";
     PXXX = SUBSTR(TD.K,1,3);
   END;
   IF SUBSTR(TV.K,1,4) ¬= "SOSS"
   THEN DO;
     DO J = 1 TO 7;
       JCLR = SP.J;
       II = POS('????????????',JCLR);
       IF II > Ø
       THEN DO;
         IF TS.K ¬= " "
         THEN JCLR = SUBSTR(JCLR,1,II-1) || 'STORAGEGROUP' ||,
                     SUBSTR(JCLR,II+12);
         ELSE JCLR = SUBSTR(JCLR,1,II-1) || '            ' ||,
                     SUBSTR(JCLR,II+12);
       END;
       II = POS('??????',JCLR);
       IF II > Ø
       THEN DO;
         JCLR = SUBSTR(JCLR,1,II-1) || TV.K || SUBSTR(JCLR,II+6);
       END;
       II = POS('????',JCLR);
       IF II > Ø
       THEN DO;
         JCLR = SUBSTR(JCLR,1,II-1) || TD.K || SUBSTR(JCLR,II+4);
       END;
       RECO.1 = JCLR;
       "EXECIO 1 DISKW DASDIN (STEM RECO.";
     END;
   END;
 END;
RETURN;
 /* PROC dump full dump dasd                                */
PROC_DUMPDASD:
 JO.1 = "//DUMPDASD JOBX ,'DUMPDASD',MSGLEVEL=(1,1),               ";
 JO.2 = "//         MSGCLASS=X,CLASS=S,TYPRUN=SCAN                 ";
 JO.3 = "//DUMMY    EXEC PGM=IEFBR14                               ";
 JO.4 = "//* DUMP DASD DISASTER RECOVERY                           ";
 SP.1 = "//SD?????? EXEC PGM=ADRDSSU,REGION=6M                     ";
 SP.2 = "//DASD     DD   DISP=SHR,UNIT=339Ø,vol=ser=??????         ";
 SP.3 = "//TAPE     DD   DSN=BKUP.??????(+1),DISP=(,CATLG,DELETE), ";
 SP.4 = "//         UNIT=(MAG,,DEFER),VOL=(,RETAIN,,Ø5),           ";
 SP.5 = "//         DCB=TRTCH=COMP,LABEL=(1,SL),EXPDT=99ØØØ        ";
 SP.6 = "//SYSPRINT DD   SYSOUT=*                                  ";
 SP.7 = "//SYSIN    DD   *                                         ";
 SP.8 = "  DUMP  FULL  INDDNAME(DASD)  OUTDDNAME(TAPE)  CANCELERROR";
 SP.9 = "  DUMP  FULL  INDDNAME(DASD)  OUTDDNAME(TAPE)  CANCELERROR";
 SPN = 9;
```

```
 "EXECIO 4 DISKW DASDFD (STEM jo.";
RETURN;
 /* PROC restore full dump dasd                              */
PROC_RESTDASD:
 JO.1 = "./         ADD  NAME=RD??????                              ";
 JO.2 = "./       ALIAS  NAME=RD@@@@@@                              ";
 JO.3 = "//RD?????? JOB  ,'RESTDASD',MSGLEVEL=(1,1),TIME=NOLIMIT,  ";
 JO.4 = "//         MSGCLASS=X,CLASS=S,TYPRUN=SCAN                  ";
 JO.5 = "//DUMMY    EXEC PGM=IEFBR14                                ";
 JO.6 = "//* REST DASD DISASTER RECOVERY                            ";
 SP.1 = "//SD?????? EXEC PGM=ADRDSSU,REGION=6M                      ";
 SP.2 = "//DASD     DD   DISP=SHR,UNIT=339Ø,VOL=SER=??????         ";
 SP.3 = "//TAPE     DD   DISP=SHR,DSN=******,                       ";
 SP.4 = "//         UNIT=(359Ø-1,,DEFER),LABEL=(???,SL),            ";
 SP.5 = "//         VOL=(,RETAIN,,1,SER=(#######))                  ";
 SQ.5 = "//         VOL=(,RETAIN,,,REF=*.SD$$$$$$.TAPE)             ";
 SP.6 = "//SYSPRINT DD   SYSOUT=*                                   ";
 SP.7 = "//SYSIN    DD   *                                          ";
 SP.8 = "  RESTORE INDDNAME(TAPE) OUTDDNAME(DASD) PURGE             ";
 SPN = 8;
 PR_MV = " ";
 PR_VL = TB_BKP_N2.1;
 DO K = 1 TO VI;
   FSN = TB_BKP_F1.K;
   BNM = TB_BKP_NM.K;
   BVS = TB_BKP_V1.K;
   DVL = TB_BKP_N2.K;
   FLG_NV = "N";
   IF BVS ¬= PR_MV
   THEN DO;
     DO J = 1 TO 6;
       JCLR = JO.J;
       II = POS('??????',JCLR);
       IF II > Ø
       THEN DO;
         JCLR = SUBSTR(JCLR,1,II-1) || BVS || SUBSTR(JCLR,II+6);
       END;
       II = POS('@@@@@@',JCLR);
       IF II > Ø
       THEN DO;
         JCLR = SUBSTR(JCLR,1,II-1) || DVL || SUBSTR(JCLR,II+6);
       END;
       RECO.1 = JCLR;
       "EXECIO 1 DISKW DASDRS (STEM RECO.";
     END;
     IF FSN > 1
     THEN DO;
       SAY FSN BVS TB_BKP_NM.K K;
       FLG_NV = "Y";
     END;
     PR_MV = BVS;
   END;
```

```
   IF BNM ¬= "?"
   THEN DO;
     DO J = 1 TO SPN;
       IF (FSN > 1 & J = 5 & SUBSTR(TB_BKP_V2.K,1,1) = '*') &,
          ^ (FLG_NV = "Y" & J = 5)
       THEN JCLR = SQ.J;
       ELSE JCLR = SP.J;
       II = POS('$$$$$$',JCLR);
       IF II > Ø
       THEN DO;
         JCLR = SUBSTR(JCLR,1,II-1) || PR_VL || SUBSTR(JCLR,II+6);
       END;
       II = POS('??????',JCLR);
       IF II > Ø
       THEN DO;
         JCLR = SUBSTR(JCLR,1,II-1) || TB_BKP_N2.K ||,
                SUBSTR(JCLR,II+6);
       END;
       II = POS('******',JCLR);
       IF II > Ø
       THEN DO;
         JCLR = SUBSTR(JCLR,1,II-1) || BNM || SUBSTR(JCLR,II+6);
       END;
       II = POS('#######',JCLR);
       IF II > Ø
       THEN DO;
         IF SUBSTR(TB_BKP_V2.K,1,1) = "*"
         THEN JCLR = SUBSTR(JCLR,1,II-1) || BVS || SUBSTR(JCLR,II+6);
         ELSE JCLR = SUBSTR(JCLR,1,II-1) || BVS ||,
                     "," || TB_BKP_V2.K || SUBSTR(JCLR,II+6);
       END;
       II = POS('???',JCLR);
       IF II > Ø
       THEN DO;
         JCLR = SUBSTR(JCLR,1,II-1) || FSN || SUBSTR(JCLR,II+3);
       END;
       RECO.1 = JCLR;
       "EXECIO 1 DISKW DASDRS (STEM RECO.";
     END;
   END;
   PR_VL = TB_BKP_N2.K;
 END;
RETURN;
 /* PROC restore full dump dasd                          */
PROC_RESTDFDS:
 JO.1 = "./         ADD  NAME=RD??????                              ";
 JO.2 = "//RD?????? JOB  ,'RESTDASD',MSGLEVEL=(1,1),                ";
 JO.3 = "//         MSGCLASS=X,CLASS=S,TYPRUN=SCAN                  ";
 JO.4 = "//DUMMY    EXEC PGM=IEFBR14                                ";
 JO.5 = "//* REST DASD DISASTER RECOVERY                            ";
 SP.1 = "//SD?????? EXEC PGM=ADRDSSU,REGION=6M                      ";
 SP.2 = "//DASD     DD   DISP=SHR,UNIT=339Ø,VOL=SER=??????         ";
```

```
SP.3 = "//TAPE     DD   DISP=SHR,DSN=******,                          ";
SP.4 = "//         UNIT=(359Ø,,DEFER),                               ";
SP.5 = "//         VOL=(,RETAIN,,1)                                  ";
SQ.5 = "//         VOL=(,RETAIN,,,REF=*.SD$$$$$$.TAPE)               ";
SP.6 = "//SYSPRINT DD   SYSOUT=*                                     ";
SP.7 = "//SYSIN    DD   *                                            ";
SP.8 = "  RESTORE INDDNAME(TAPE) OUTDDNAME(DASD) PURGE               ";
SPN = 8;
PR_MV = " ";
PR_VL = TB_BKP_N2.1;
DO K = 1 TO VI;
  FSN = TB_BKP_F1.K;
  BNM = TB_BKP_NM.K;
  BVS = TB_BKP_V1.K;
  FLG_NV = "N";
  IF BVS ¬= PR_MV
  THEN DO;
    DO J = 1 TO 5;
      JCLR = JO.J;
      II = POS('??????',JCLR);
      IF II > Ø
      THEN DO;
        JCLR = SUBSTR(JCLR,1,II-1) || BVS || SUBSTR(JCLR,II+6);
      END;
      RECO.1 = JCLR;
      "EXECIO 1 DISKW DASDRS (STEM RECO.";
    END;
    IF FSN > 1
    THEN DO;
      SAY FSN BVS TB_BKP_NM.K K;
      FLG_NV = "Y";
    END;
    PR_MV = BVS;
  END;
  IF BNM ¬= "?"
  THEN DO;
    DO J = 1 TO SPN;
      IF (FSN > 1 & J = 5 & SUBSTR(TB_BKP_V2.K,1,1) = '*') &,
         ^ (FLG_NV = "Y" & J = 5)
      THEN JCLR = SQ.J;
      ELSE JCLR = SP.J;
      II = POS('$$$$$$',JCLR);
      IF II > Ø
      THEN DO;
        JCLR = SUBSTR(JCLR,1,II-1) || PR_VL || SUBSTR(JCLR,II+6);
      END;
      II = POS('??????',JCLR);
      IF II > Ø
      THEN DO;
        JCLR = SUBSTR(JCLR,1,II-1) || TB_BKP_N2.K ||,
               SUBSTR(JCLR,II+6);
      END;
```

```
       II = POS('******',JCLR);
       IF II > Ø
       THEN DO;
         JCLR = SUBSTR(JCLR,1,II-1) || BNM || SUBSTR(JCLR,II+6);
       END;
       II = POS('######',JCLR);
       IF II > Ø
       THEN DO;
         IF SUBSTR(TB_BKP_V2.K,1,1) = "*"
         THEN JCLR = SUBSTR(JCLR,1,II-1) || BVS || SUBSTR(JCLR,II+6);
         ELSE JCLR = SUBSTR(JCLR,1,II-1) || BVS ||,
                     "," || TB_BKP_V2.K || SUBSTR(JCLR,II+6);
       END;
       II = POS('???',JCLR);
       IF II > Ø
       THEN DO;
         JCLR = SUBSTR(JCLR,1,II-1) || FSN || SUBSTR(JCLR,II+3);
       END;
       RECO.1 = JCLR;
       "EXECIO 1 DISKW DASDRS (STEM RECO.";
     END;
   END;
   PR_VL = TB_BKP_N2.K;
 END;
RETURN;
```

REXX PROC to create logical dump of catalogs:

```
/* REXX                                                               */
/*--------------------------------------------------------------------*/
/* Proc drpxLCAT                                                      */
/*--------------------------------------------------------------------*/
/* LISTCAT UCAT AND BUILD  DFDSS DUMP COMMANDS                        */
 'PROFILE NOPREFIX';
 X = OUTTRAP('RECL.');
 "LISTCAT UCAT";
 SAY 'THE NUMBER OF RECORDS TRAPPED IS' RECL.Ø;
 DSN_NM = '';
 RECO.1 = "  DUMP DS(INC( -";
 DO I = 1 TO RECL.Ø;
   K = I + 1;
   RECO.K = "    " || WORD(RECL.I,3) || " , -";
 END;
 K = K + 1;
 RECO.K = "         ))   ADMIN        OUTDD(OUT) ";
 "EXECIO" K " DISKW DUMPCAT (STEM RECO.";
 "EXECIO Ø DISKW DUMPCAT (FINIS";
 RETURN;
```

REXX PROC to eject our last volumes:

```
/* REXX                                                               */
```

```
/*-------------------------------------------------------------------*/
/* Proc drpxejeØ                                                     */
/*   Input : BKPA   -> generic mak for backup datasets               */
/*-------------------------------------------------------------------*/
 ARG BKPA NBRD NBRL BLM
 TRACE o;
 CALL PROC_PARM;
 CALL PROC_NONVSFL;
 CALL PROC_EJECTVL;
RETURN;
 /* Proc parm                                                 */
PROC_parm:
 IF BKPA = "" THEN HQNVS = "BKUP.SOSS28.G*";
              ELSE HQNVS = BKPA;
 IF NBRD = "" THEN NBRD = 7;
 IF NBRL = "" THEN NBRL = 21;
 IF BLM  = "" THEN BLM  = 99;
 DATE_WKJ = DATE('J');
 DATE_WKS = DATE('S');
 YEAR_BKUP = SUBSTR(DATE_WKS,1,4);
 DAY_BKUP = SUBSTR(DATE_WKJ,3,3) - NBRD;
 IF DAY_BKUP < Ø
 THEN DO;
   DAY_BKUP = 365 - DAY_BKUP;
   YEAR_BKUP = YEAR_BKUP - 1;
 END;
 DAY_BKUP = RIGHT(DAY_BKUP,3,"Ø");
 BKUPD = YEAR_BKUP || DAY_BKUP;
 YEAR_BKUP = SUBSTR(DATE_WKS,1,4);
 DLM_BKUP = SUBSTR(DATE_WKJ,3,3) - NBRL;
 IF DLM_BKUP < Ø
 THEN DO;
   DLM_BKUP = 365 - DLM_BKUP;
   YEAR_BKUP = YEAR_BKUP - 1;
 END;
 Dlm_BKUP = RIGHT(Dlm_BKUP,3,"Ø");
 BKUPl = YEAR_BKUP || Dlm_BKUP;
 say " ****************************************** ";
 SAY "   nbrd : " nbrd ;
 SAY "   date : " date_wks;
 SAY "   date : " date_wkj;
 SAY "   MinD : " DAY_BKUP;
 SAY "   MaxD : " DLM_BKUP;
 SAY "   last : " BKUPD;
 SAY "   old  : " BKUPl;
 SAY "   BkVer: " BLM;
 say " ****************************************** ";
RETURN;
 /* Proc NonVS_fl                                             */
PROC_NONVSFL:
 KEY = HQNVS || '.**';
```

```
COUNT = Ø                           /* TOTAL ENTIRES FOUND          */
MODRSNRC = SUBSTR(' ',1,4)          /*   CLEAR MODULE/RETURN/REASON */
CSIFILTK = SUBSTR(KEY,1,44)         /*   MOVE FILTER KEY INTO LIST  */
CSICATNM = SUBSTR(' ',1,44)         /*   CLEAR CATALOG NAME         */
CSIRESNM = SUBSTR(' ',1,44)         /*   CLEAR RESUME NAME          */
CSIDTYPS = SUBSTR('ABH',1,16)       /*   CLEAR ENTRY TYPES          */
CSICLDI  = SUBSTR('Y',1,1)          /*   INDICATE DATA AND INDEX    */
CSIRESUM = SUBSTR(' ',1,1)          /*   CLEAR RESUME FLAG          */
CSIS1CAT = SUBSTR(' ',1,1)          /* SEARCH > 1 CATALOGS          */
CSIRESRV = SUBSTR(' ',1,1)          /*   CLEAR RESERVE CHARACTER    */
CSINUMEN = 'ØØØ5'X                  /*   INIT NUMBER OF FIELDS      */
CSIFLD1    = SUBSTR('VOLSER',1,8)   /*   INIT FIELD 1 FOR VOLSERS   */
CSIFLD2    = SUBSTR('DEVTYP',1,8)   /*   INIT FIELD 2 FOR DEVTYP    */
CSIFLD3    = SUBSTR('FILESEQ',1,8)  /*   INIT FIELD 5 FOR DS EX DT  */
CSIFLD4    = SUBSTR('DSCRDT2',1,8)  /*   INIT FIELD 3 FOR DS CR DT  */
CSIFLD5    = SUBSTR('DSEXDT2',1,8)  /*   INIT FIELD 4 FOR DS EX DT  */
 /*  BUILD THE SELECTION CRITERIA FIELDS PART OF PARAMETER LIST     */
CSIOPTS  = CSICLDI || CSIRESUM || CSIS1CAT || CSIRESRV
CSIFIELD = CSIFILTK || CSICATNM || CSIRESNM || CSIDTYPS || CSIOPTS
CSIFIELD = CSIFIELD || CSINUMEN || CSIFLD1 || CSIFLD2 || CSIFLD3
CSIFIELD = CSIFIELD || CSIFLD4 || CSIFLD5;
 /*  INITIALIZE AND BUILD WORK ARE OUTPUT PART OF PARAMETER LIST    */
WORKLEN = 131Ø72  /* 128K */
DWORK = 'ØØØ2ØØØØ'X || COPIES('ØØ'X,WORKLEN-4)
 /*  INITIALIZE WORK VARIABLES                                      */
RESUME = 'Y'
CATNAMET = SUBSTR(' ',1,44)
DNAMET = SUBSTR(' ',1,44)
IC = Ø;
 /*  SET UP LOOP FOR RESUME (IF A RESUME IS NECESSARY)              */
DO WHILE RESUME = 'Y'
 /*  ISSUE LINK TO CATALOG GENERIC FILTER INTERFACE                 */
  ADDRESS LINKPGM 'IGGCSIØØ  MODRSNRC  CSIFIELD  DWORK'
  RESUME = SUBSTR(CSIFIELD,15Ø,1);
  USEDLEN = C2D(SUBSTR(DWORK,9,4));
  POS1=15;
 /*  PROCESS DATA RETURNED IN WORK AREA                             */
  DO WHILE POS1 < USEDLEN
    IF SUBSTR(DWORK,POS1+1,1) = 'Ø'
    THEN DO
      CATNAME=SUBSTR(DWORK,POS1+2,44)
      POS1 = POS1 + 5Ø
    END
    IF POS1 < USEDLEN    /* IF STILL MORE DATA       */
    then DO              /* CONTINUE WITH NEXT ENTRY */
      DNAME = SUBSTR(DWORK,POS1+2,44)  /* GET ENTRY NAME   */
 /*  ASSIGN ENTRY TYPE NAME                                         */
      SELECT;
        WHEN SUBSTR(DWORK,POS1+1,1) = 'C' THEN DTYPE = 'CLUSTER '
        WHEN SUBSTR(DWORK,POS1+1,1) = 'D' THEN DTYPE = 'DATA    '
        WHEN SUBSTR(DWORK,POS1+1,1) = 'I' THEN DTYPE = 'INDEX   '
        WHEN SUBSTR(DWORK,POS1+1,1) = 'A' THEN DTYPE = 'NONVSAM '
```

```
        WHEN SUBSTR(DWORK,POS1+1,1) = 'H' THEN DTYPE = 'GDS     '
        WHEN SUBSTR(DWORK,POS1+1,1) = 'B' THEN DTYPE = 'GDG     '
        WHEN SUBSTR(DWORK,POS1+1,1) = 'R' THEN DTYPE = 'PATH    '
        WHEN SUBSTR(DWORK,POS1+1,1) = 'G' THEN DTYPE = 'AIX     '
        WHEN SUBSTR(DWORK,POS1+1,1) = 'X' THEN DTYPE = 'ALIAS   '
        WHEN SUBSTR(DWORK,POS1+1,1) = 'U' THEN DTYPE = 'UCAT    '
        OTHERWISE DO;
          IF SUBSTR(DWORK,POS1+1,1) = '0' THEN ITERATE
          POS1 = POS1 + 46
          POS1 = POS1 + C2D(SUBSTR(DWORK,POS1,2))
          ITERATE
        END;
      end;
/* CHECK FOR ERROR IN ENTRY RETURNED                              */
      CSIEFLAG = SUBSTR(DWORK,POS1+0,1)
      IF BITAND(CSIEFLAG,'40'X) = '40'X
      then do
        POS1 = POS1 + 50           /* HEADER LENGTH */
        POS1 = POS1 + C2D(SUBSTR(DWORK,POS1,2))
        ITERATE    /* GO TO NEXT ENTRY */
      END
/*  HAVE NAME AND TYPE, GET VOLSER INFO                                */
      COUNT = COUNT + 1  /* total entires found  */
      POS1 = POS1 + 46
      IF DTYPE = 'NONVSAM' | DTYPE = 'GDS'
      THEN DO;
        NUMVOL = C2D(SUBSTR(DWORK,POS1+4,2))/6
        DATAPV = 14;                 /* POINTER TO FIRST VOLSER  */
        DATAPD = 14 + (NUMVOL * 6);  /* POINTER TO FIRST DEVTYPE */
        DATAPF = 14 + (NUMVOL * 10); /* POINTER TO FIRST FILESEQ */
        DATAPX = 14 + (NUMVOL * 12); /* POINTER TO OTHER DATA    */
                               /* 12 = 6 VOL + 4 DATE + 2 FILENO */
        POS2 = POS1 + 4 + (NUMVOL * 12) + 10;
        POSD = POS2 + (NUMVOL * 6) + 10;
        POSF = POSD + (NUMVOL * 4) + 10;
        VOL2 = copies('*',6);
        VOL3 = copies('*',6);
        DVT2 = copies('*',8);
        DVT3 = copies('*',8);
        FSN2 = '***';
        FSN3 = '***';
      /* SAY "DEBUG00:"SUBSTR(DWORK,POS1,80); */
        VOL1 = left(substR(DWORK,POS1+DATAPV,6),6," ");
        DVT1 = C2X(SUBSTR(DWORK,POS1+DATAPD,4));
        FSN1 = RIGHT(C2D(SUBSTR(DWORK,POS1+DATAPF,2)),3,"0");
        IF NUMVOL > 1
        THEN DO;
          VOL2 = SUBSTR(DWORK,POS1+DATAPV+6,6);
          DVT2 = C2X(SUBSTR(DWORK,POS1+DATAPD+4,4));
          FSN2 = RIGHT(C2D(SUBSTR(DWORK,POS1+DATAPF+2,2)),3,"0");
        END;
        IF NUMVOL > 2
```

```
          THEN DO;
            VOL3 = SUBSTR(DWORK,POS1+DATAPV+12,6);
            DVT3 = C2X(SUBSTR(DWORK,POS1+DATAPD+8,4));
            FSN3 = RIGHT(C2D(SUBSTR(DWORK,POS1+DATAPF+4,2)),3,"Ø");
          END;
          CRDT = C2X(SUBSTR(DWORK,POS1+DATAPX,4));
          CDDD = SUBSTR(CRDT,3,3);
          CYY = SUBSTR(CRDT,1,2);
          CCC = SUBSTR(CRDT,7,2);
          IF CCC = 'ØØ' THEN CCC = '19';
                       ELSE CCC = '2Ø';
          CRDT = CCC || CYY || CDDD;
          IC = IC + 1;
          RECI.IC = DNAME NUMVOL CRDT, /* EXDT, */
                    VOL1 VOL2 VOL3,
                    FSN1 FSN2 FSN3,
                    DVT1 DVT2 DVT3;
        END;
  /*    GET POSITION OF NEXT ENTRY                                    */
        POS1 = POS1 + C2D(SUBSTR(DWORK,POS1,2))
      END;
    END;
 END;
RETURN;
 /* PROC EJECT VOLUME                                    */
PROC_EJECTVL:
 DO I = 1 TO IC;
   BKP_NM = WORD(RECI.I,1);
   BKP_V1 = WORD(RECI.I,4);
   SAY BKP_NM BKP_V1;
 END;
 IF IC > Ø
 THEN DO;
   ADDRESS TSO "CONSOLE ACTIVATE"
   ADDRESS CONSOLE "CART DRPEJECT"
   COMMAND = "D SMS,LIB(LIBVTS),DETAIL";
   COMMAND = "LI E,"||BKP_V1;
   ADDRESS CONSOLE  COMMAND
   GETCODE = GETMSG('PRTMSG.','SOL','DRPEJECT',,6Ø)
   IF GETCODE = Ø
   THEN
     DO I = 1 TO PRTMSG.Ø
       SAY PRTMSG.I
     END
   ELSE  SAY "GETMSG ERROR RETRIEVING MESSAGE.  RETURN CODE IS" GETCODE
   ADDRESS TSO "CONSOLE DEACTIVATE"
 end;
RETURN;
```

Alain Piraux
System Engineer (Belgium)

WANT TO SUBSCRIBE?

Each monthlyly issue of *MVS Update* is packed with ideas for improving *your* MVS installation – and all of the technical details necessary for *you* to put those ideas into practice.

With *MVS Update* ***you*** get:

- A practical toolkit of ready-made enhancements, developed and tested in a working environment by MVS experts throughout the world.
- Tutorial articles on system internals.
- Performance tuning tips and measurements.
- Early user reports on new products and releases.

Plus:

- *MVS Update* will repay the cost of subscribing many times over!

All for a fraction of the cost of a single training course! So what are *you* waiting for?

- Go to http://www.xephonUSA.com/subscribe now to subscribe!
- Subscribe now and receive 25% off of a 12-month subscription*!

***** using promotional code **MV8X42**.

* * *

WANT TO CONTRIBUTE?

MVS Update is written by technical professionals just like you with a desire to share their expert knowledge with the world.

Xephon is always seeking talented individuals to contribute articles to *MVS Update* – **and get paid for it!**

If you have insight into how to make MVS more functional, secure, reliable, user friendly, or to generally improve MVS performance, please visit http://www.xephonUSA.com/contribute for more details on how you can contribute to this definitive industry publication.