210

MVS update

March 2004

In this issue

MVS Update

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

North American office
Xephon
PO Box 350100
Westminster, CO 80035-0100
USA
Telephone: 303 410 9344

Subscriptions and back-issues
A year's subscription to *MVS Update*, comprising twelve monthly issues, costs £340.00 in the UK; $505.00 in the USA and Canada; £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 1999 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.

Editor
Trevor Eddolls

Contributions
When Xephon is given copyright, articles published in *MVS Update* are paid for at the rate of £100 ($160) per 1000 words and £50 ($80) per 100 lines of code for the first 200 lines of original material. The remaining code is paid for at the rate of £20 ($32) 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.

Checking tape volids

We received a delivery of pre-initialized and pre-labelled 3490-type cartridges. Unfortunately, the internal and external labels did not match in all cases! I wrote the following program to cross-check tape volids. Sample JCL is included in the program.

Output for a mismatch (via WTO, also gives RC=4):

```
 +============================================================
 +CHKCARTS: MISMATCH, EXTERNAL LABEL="7Ø24Ø4", VOL1="7Ø2434"
 +============================================================
```

Output for a match (also gives RC=0):

```
 +CHKCARTS: CART "7Ø24Ø4" VERIFIED OK...
```

PROGRAM CODE

```
CHKCARTS TITLE 'CHECK VOL1 LABELS AGAINST EXTERNAL LABELS'
***********************************************************************
*    MODULE:   CHKCARTS                                               *
*    DESC:     READ A CART BLP AND VERIFY THAT THE VOL1 MATCHES       *
*              WHAT WE ARE EXPECTING...                               *
*                                                                     *
*              THIS IS BECAUSE PRE-INITIALIZED CARTRIDGES WERE        *
*              DELIVERED TO US WITH THE INTERNAL/EXTERNAL LABELS      *
*              NOT MATCHING.                                          *
*    JCL:                                                             *
*           -------------------------------------------------------   *
*          ¦   //jobcard                                          ¦   *
*          ¦   //S1      EXEC PGM=CHKCARTS                        ¦   *
*          ¦   //STEPLIB  DD  DSN=<user.loadlib>,DISP=SHR         ¦   *
*          ¦   //SYSPRINT DD  SYSOUT=*                            ¦   *
*          ¦   //INPUT    DD  DSN=anyname,UNIT=349Ø,              ¦   *
*          ¦   //            VOL=SER=nnnnnn,LABEL=(1,BLP),        ¦   *
*          ¦   //            EXPDT=98ØØØ,RECFM=U,BLKSIZE=4Ø96     ¦   *
*           -------------------------------------------------------   *
*                                                                     *
***********************************************************************
         PRINT NOGEN
*---------------------------------------------------------------------*
* HOUSEKEEPING, ETC...                                                *
*---------------------------------------------------------------------*
CHKCARTS CSECT
         BAKR  R14,Ø                   SAVE CALLER DATA ON STACK
         LR    R12,R15                  GET ENTRY POINT
         USING CHKCARTS,R12             ADDRESSABILITY TO MODULE
*---------------------------------------------------------------------*
* GET JFCB INFORMATION FOR LATER CHECKING...                          *
*---------------------------------------------------------------------*
         RDJFCB INPUT                   GET JFCB FOR INPUT DATASET
         MVC   VOLXJFCB,JFCB+118        SAVE VOLSER FROM JFCB
*
         OPEN  INPUT
*
*---------------------------------------------------------------------*
* READ THE VOL1 FROM THE CART AND VERIFY THAT THE VOLSER ON THE CART  *
* MATCHES THAT SPECIFIED IN THE JCL...                                *
*---------------------------------------------------------------------*
READVOL1 DS    ØH
         GET   INPUT                    READ VOL1
         LR    R2,R1                    GET ADDRESS OF VOL1
         CLC   Ø(3,R2),=C'VOL1'         IS IT A VOL1?
         BNE   BADVOL1                  NO...
         MVC   VOLSER,4(R2)             .....SAVE VOLSER
         CLC   VOLSER,VOLXJFCB          VOLSER IN HDR1 SAME AS IN JCL?
         BNE   MISMATCH                 NO...OH DEAR...
         MVC   OKWTO+24(6),VOLSER       SHOW ITS OK
OKWTO    WTO   'CHKCARTS: CART "......" VERIFIED OK...',               X
               ROUTCDE=11
*---------------------------------------------------------------------*
* GET OUT OF HERE...                                                  *
*---------------------------------------------------------------------*
RETURN   DS    ØH
         CLOSE (INPUT)
         L     R15,RETC                 SET RC
         PR    ,                        RETURN
*---------------------------------------------------------------------*
* COULD NOT IDENTIFY VOL1 RECORD...                                   *
*---------------------------------------------------------------------*
BADVOL1  DS    ØH
         WTO   MF=(E,WTOHDR),ROUTCDE=11
         MVC   WTOVOL1+26(47),Ø(R2)    MOVE RECORD TO WTO
         WTO   MF=(E,WTOVOL1),ROUTCDE=11
         WTO   MF=(E,WTOHDR),ROUTCDE=11
         B     RETURN
*---------------------------------------------------------------------*
* EITHER DSNAME OR VOLSER SPECIFIED IN THE JCL DOES NOT MATCH WHAT IS *
* IN THE HDR1...                                                      *
*---------------------------------------------------------------------*
MISMATCH DS    ØH
         WTO   MF=(E,WTOHDR),ROUTCDE=11
         MVC   WTOMISS+4Ø(6),VOLXJFCB  MOVE JFCB VOL TO WTO
         MVC   WTOMISS+55(6),VOLSER    MOVE VOL1 VOL TO WTO
         WTO   MF=(E,WTOMISS),ROUTCDE=11
         WTO   MF=(E,WTOHDR),ROUTCDE=11
         MVC   RETC,RC4                SET RC=4
         B     RETURN
*---------------------------------------------------------------------*
* E-O-V EXIT: SHOULD NEVER COME HERE, SO ISSUE MSG AND ABEND...       *
*---------------------------------------------------------------------*
EOVEXIT  DS    ØH
         LR    R3,R15                  GET ENTRY POINT ADDRESS
         USING *,R3                    ADDRESSABILITY TO EXIT
         WTO   MF=(E,WTOHDR),ROUTCDE=11
         WTO   'CHKCARTS: E-O-V EXIT DRIVEN...ABENDING',ROUTCDE=11
         WTO   MF=(E,WTOHDR),ROUTCDE=11
         DS    F                       RETURN FROM EXIT
*---------------------------------------------------------------------*
* E-O-F REACHED: SHOULD NEVER COME HERE, SO ISSUE MSG AND ABEND...    *
*---------------------------------------------------------------------*
BADEOF   DS    ØH
         LR    R3,R15                  GET ENTRY POINT ADDRESS
         USING *,R3                    ADDRESSABILITY TO EXIT
         WTO   MF=(E,WTOHDR),ROUTCDE=11
         WTO   'CHKCARTS: E-O-F REACHED...ABENDING',ROUTCDE=11
         WTO   MF=(E,WTOHDR),ROUTCDE=11
         DS    F                       RETURN FROM EXIT
         DROP  R3
*---------------------------------------------------------------------*
*        CONSTANTS                                                    *
*---------------------------------------------------------------------*
*
         YREGS
*
         DS    ØD
         DC    CL16'***EYECATCHER***'
VOLSER   DC    CL6' '
VOLXJFCB DC    CL6' '
RETC     DC    F'Ø'
RC4      DC    F'4'
*
INPUT    DCB   DDNAME=INPUT,                                           X
               DSORG=PS,                                               X
               MACRF=GL,                                               X
               EXLST=EXLSTIN,                                          X
               EODAD=BADEOF
*
EXLSTIN  DS    ØF
         DC    X'Ø7'                   SHOWS THIS IS A READ JFCB EXIT
         DC    AL3(JFCB)               ADDRESS OF JFCB AREA
         DC    X'86'                   SHOWS THIS IS AN E-O-V EXIT
         DC    AL3(EOVEXIT)            ADDRESS OF E-O-V EXIT
JFCB     DC    176C' '
*
```

```
WTOHDR   WTO   '=======================================================X
               ====',ROUTCDE=11,MF=L
WTOVOL1  WTO   'CHKCARTS: DODGY VOL1 "................................X
               ..............."',ROUTCDE=11,MF=L
WTOMISS  WTO   'CHKCARTS: MISMATCH, EXTERNAL LABEL="......", VOL1="....X
               .."',ROUTCDE=11,MF=L
*
         END
```

Grant Carson
Systems Programmer (UK)

Mathematical functions for REXX

As a long-time REXX lover, I have developed many programs for a thousand and one purposes both in VM and MVS. Adding a few mathematical functions to it was not only entertaining, it also made me use methods learned at school that tend to get forgotten over time (through lack of use). It also provided an opportunity to do a bit of research on other topics less familiar to me (like hyperbolic functions).

The result is this collection of eighteen functions, which include square root (SQR), cubic root (CUR), natural logarithm (LN) and its inverse exponent (EXP), decimal logarithm (LOG), factorial (FACT), the base trigonometric functions (SIN, COS, TAN), their inverse ‘arc’ functions (ASIN, ACOS, ATAN), and the hyperbolic functions – both direct (SINH, COSH, TANH) and inverse (ASINH, ACOSH, ATANH).

Some are written as pure numerical methods, like SQR or LN, some are based on other functions, like the hyperbolics, which are derived from simple expressions containing EXP, LN, and SQR; others are a mixture of both, like SIN and COSIN, which use the FACT function within an iteration.

The ATAN function uses a mixture method – since near 1 (or 45 degrees) the iteration method would require a large number of passes to produce accurate results. For input values between

0.8 and 1.2 (corresponding to answers 38 to 50 degrees approximately) I use an alternative method known as Taylor expansion. Since the coefficients involved in this method are independent of the input, they can be calculated beforehand and thus I wrote them directly in the expressions. The number of coefficients used is enough to guarantee a degree of precision identical to the iterative method used for values outside the stated interval.

On the subject of precision, in most cases the number of accurate digits is at least 19. However, on some functions that use other function results (like tangent, defined as sin / cos), this number can be slightly smaller; dropping to 18, or 17 in extreme cases.

All trigonometric functions have their input or output in degrees. If you want to work with radians, you must do the appropriate conversion yourself.

Any invalid input for a function – either not a valid number or a number outside the function domain – will return the string 'Error'.

All functions can be called from within other functions or programs, or directly from a terminal prompt, passing them the argument. In the first case, the answer is returned to the caller, and in the second case it is displayed on the terminal.

NATURAL LOGARITHM FUNCTION

```
/*= REXX ============================================================*/
/*  LN                                                               */
/*  Natural logarithm function.                                      */
/*===================================================================*/
arg x .
if datatype(x,"N") = Ø | x < Ø | x = Ø then do
   y = "Error"
   signal saida
end
if x = 1 then do
   y = Ø
   signal saida
```

```
end
numeric digits 2Ø
neg = ""
if x < 1 then do
   x = 1/x
   neg = "-"
end
do f = 1 to 1ØØØ while x > 5
   x = x/5
end
f = f - 1
if f = Ø then y = lneper(x)
else y = lneper(x) + lneper(5) * f
y = neg||y
saida:
 parse source . calltype .
 if calltype = "COMMAND" then say y
 else return y
exit
/*==============*/
lneper: procedure
 arg z
 k = (z-1)/(z+1)
 v = k
 do j = 3 to 99 by 2
    v1 = v + (k**j)/j
    if v1 = v then leave
    v = v1
 end
return v*2
```

DECIMAL LOGARITHM FUNCTION

```
/*= REXX ============================================================*/
/*  LOG                                                              */
/*  Decimal logarithm function.                                      */
/*===================================================================*/
arg x .
if datatype(x,"N") = Ø | x < Ø | x = Ø then do
   y = "Error"
   signal saida
end
if x = 1 then do
   y = Ø
   signal saida
end
numeric digits 2Ø
y = ln(x)/ln(1Ø)
saida:
 parse source . calltype .
 if calltype = "COMMAND" then say y
 else return y
exit
```

EXPONENT FUNCTION (INVERSE NATURAL LOGARITHM)

```
/*= REXX ==================================================================*/
/*   EXP                                                                   */
/*   Exponent function (inverse of natural logarithm)                      */
/*=========================================================================*/
arg x .
if datatype(x,"N") = Ø then do
   y = "Error"
   signal saida
end
numeric digits 2Ø
if x < Ø then do
   x = abs(x)
   inverse = 1
end
do f = 1 to 1ØØØ while x > 5
   x = x - 5
end
f = f - 1
if f > Ø then y = exponent(x)*exponent(5)**f
else y = exponent(x)
if inverse = 1 then y = 1/y
saida:
 parse source . calltype .
 if calltype = "COMMAND" then say y
 else return y
exit
/*=================*/
exponent: procedure
 arg j
 b = 1
 do k = 1 to 1ØØ
    b1 = b + (j**k)/fact(k)
    if b = b1 then leave
    b = b1
 end
return b
```

FACTORIAL FUNCTION

```
/*= REXX ==================================================================*/
/*   FACT                                                                  */
/*  Factorial function.                                            */
/*=================================================================*/
arg x .
if datatype(x,"W") = Ø |  x < Ø then do
   y = "Error"
   signal saida
end
numeric digits 2Ø
y = 1
do k = 1 to x
   y = y*k
end
saida:
 parse source . calltype .
 if calltype = "COMMAND" then say y
 else return y
exit
```

SQUARE ROOT FUNCTION

```
/*= REXX ==========================================================*/
/*  SQR                                                            */
/*  Square root function.                                          */
/*=================================================================*/
arg x .
if  datatype(x,"N") = Ø | x < Ø then do
   y = "Error"
   signal saida
end
if x = Ø then do
   y = Ø
   signal saida
end
numeric digits 2Ø
y = x
do 2ØØ
   y1 = (y+x/y)/2
   if y1 = y then leave
   y = y1
end
saida:
 parse source . calltype .
 if calltype = "COMMAND" then say y
 else return y
exit
```

CUBIC ROOT FUNCTION

```
/*= REXX ==========================================================*/
```

```
/*  CUR                                                              */
/*  Cubic root function.                                             */
/*===================================================================*/
arg x .
if datatype(x,"N") = Ø then do
   y = "Error"
   signal saida
end
if x = Ø then do
   y = Ø
   signal saida
end
numeric digits 2Ø
y = x
do 5ØØ
   y1 = (2/3)*y + (1/3)*(x/(y**2))
   if y1 = y then leave
   y = y1
end
saida:
 parse source . calltype .
 if calltype = "COMMAND" then say y
 else return y
exit
```

SINE FUNCTION

```
/*= REXX ============================================================*/
/*  SIN                                                              */
/*  Sine function. Input must be degrees.                            */
/*===================================================================*/
arg x .
if datatype(x,"N") = Ø | abs(x) > 1ØØØØ then do
   y = "Error"
   signal saida
end
numeric digits 2Ø
pi = "3.14159265358979323846"
pi2 = pi*2
x = x*pi/18Ø
do forever
   if abs(x) < pi2 then leave
   if x > Ø then x = x - pi2
  else x = x + pi2
end
y = Ø
do k = 1 to 4ØØ by 4
   y1 = y + (x**k)/fact(k)
   j  = k+2
   y1 = y1 - (x**j)/fact(j)
   if y = y1 then leave
   y = y1
end
if abs(y) < 1e-18 then y = Ø
saida:
 parse source . calltype .
 if calltype = "COMMAND" then say y
 else return y
exit
```

COSINE FUNCTION

```
/*= REXX ============================================================*/
/*  COS                                                              */
/*  Cosine function.Input must be degrees.                           */
/*===================================================================*/
arg x .
if datatype(x,"N") = Ø | abs(x) > 1ØØØØ then do
   y = "Error"
   signal saida
end
numeric digits 2Ø
pi = "3.14159265358979323846"
pi2 = pi*2
x = x*pi/18Ø
do forever
   if abs(x) < pi2 then leave
   if x > Ø then x = x - pi2
  else x = x + pi2
end
y = Ø
do k = Ø to 4ØØ by 4
   y1 = y + (x**k)/fact(k)
   j  = k+2
   y1 = y1 - (x**j)/fact(j)
   if y = y1 then leave
   y = y1
end
if abs(y) < 1e-18 then y = Ø
saida:
 parse source . calltype .
 if calltype = "COMMAND" then say y
 else return y
exit
```

TANGENT FUNCTION

```
/*= REXX ============================================================*/
```

```
/*  TAN                                                               */
/*  Tangent function. Input must be degrees.                          */
/*====================================================================*/
arg x .
if datatype(x,"N") = Ø | abs(x) > 1ØØØØ then do
   y = "Error"
   signal saida
end
numeric digits 2Ø
c = cos(x)
if abs(c) < 1e-1Ø then do
   y = "Error"
   signal saida
end
y = sin(x)/c
saida:
 parse source . calltype .
 if calltype = "COMMAND" then say y
 else return y
exit
```

ARC SINE FUNCTION

```
/*= REXX =============================================================*/
/*  ASIN                                                              */
/*  Arc Sine function.  Answer is in degrees.                         */
/*====================================================================*/
arg x .
if datatype(x,"N") = Ø | abs(x) > 1 then do
   y = "Error"
   signal saida
end
numeric digits 2Ø
if abs(x) = 1 then do
   y = x*9Ø
end
else do
   y = atan(x/sqr(1-x**2))
end
saida:
 parse source . calltype .
 if calltype = "COMMAND" then say y
 else return y
exit
```

ARC COSINE FUNCTION

```
/*= REXX =============================================================*/
```

```
/*  ACOS                                                             */
/*  Arc Cosine function.  Answer is in degrees.                      */
/*=================================================================*/
arg x .
if datatype(x,"N") = Ø | abs(x) > 1 then do
   y = "Error"
   signal saida
end
numeric digits 2Ø
if x = Ø then do
   y = 9Ø
end
else do
   y = atan(sqr(1-x**2)/x)
end
saida:
 parse source . calltype .
 if calltype = "COMMAND" then say y
 else return y
exit
```

ARC TANGENT FUNCTION

```
/*= REXX ==========================================================*/
/*  ATAN                                                             */
/*  Arc tangent function.  Answer is in degrees.                     */
/*=================================================================*/
arg x .
if datatype(x,"N") = Ø then do
   y = "Error"
   signal saida
end
numeric digits 2Ø
pi = "3.14159265358979323846"
negative = ""
if x < Ø then do
   x = abs(x)
   negative = "-"
end
if x > Ø.8 & x < 1.2 then do
   d = x-1
   y = pi/4 + Ø.5*d - Ø.25*d**2 + (1/12)*d**3 - Ø.Ø25*d**5
   y = y + (1/48)*d**6 - (1/112)*d**7 + (1/288)*d**9
   y = y - (1/32Ø)*d**1Ø + (1/7Ø4)*d**11 - (1/1664)*d**13
   y = y + (1/1792)*d**14 - (1/384Ø)*d**15
end
else do
   y = Ø
   if x < 1 then do
```

```
      do k = 1 to 5ØØ by 4
         y1 = y + (x**k)/k
         j  = k+2
         y1 = y1 - (x**j)/j
         if y = y1 then leave
         y = y1
      end
   end
   else do
      x = 1/x
      do k = 1 to 5ØØ by 4
         y1 = y - (x**k)/k
         j  = k+2
         y1 = y1 + (x**j)/j
         if y = y1 then leave
         y = y1
      end
      y = pi/2 + y
   end
end
y = negative || y*18Ø/pi
saida:
 parse source . calltype .
 if calltype = "COMMAND" then say y
 else return y
exit
```

HYPERBOLIC SINE

```
/*= REXX ===========================================================*/
/*  SINH degrees                                                    */
/*  Hyperbolic Sine. Input must be degrees.                         */
/*==================================================================*/
arg x .
if datatype(x,"N") = Ø then do
   y = "Error"
   signal saida
end
numeric digits 2Ø
y = (exp(x) - exp(-x))/2
saida:
 parse source . calltype .
 if calltype = "COMMAND" then say y
 else return y
exit
```

HYPERBOLIC COSINE

```
/*= REXX ===========================================================*/
```

```
/*   COSH degrees                                                    */
/*   Hyperbolic Cosine. Input must be degrees.                       */
/*===================================================================*/
arg x .
if datatype(x,"N") = Ø then do
   y = "Error"
   signal saida
end
numeric digits 2Ø
y = (exp(x) + exp(-x))/2
saida:
 parse source . calltype .
 if calltype = "COMMAND" then say y
 else return y
exit
```

HYPERBOLIC TANGENT

```
/*= REXX ============================================================*/
/*   TANH degrees                                                    */
/*   Hyperbolic Tangent. Input must be degrees.                      */
/*===================================================================*/
arg x .
if datatype(x,"N") = Ø then do
   y = "Error"
   signal saida
end
numeric digits 2Ø
z = exp(x)
y = (z - 1/z)/(z + 1/z)
saida:
 parse source . calltype .
 if calltype = "COMMAND" then say y
 else return y
exit
```

INVERSE HYPERBOLIC SINE

```
/*= REXX ============================================================*/
/*   ASINH                                                           */
/*   Inverse Hyperbolic Sin.  Answer is in degrees.                  */
/*===================================================================*/
arg x .
if datatype(x,"N") = Ø then do
   y = "Error"
   signal saida
end
numeric digits 2Ø
```

```
y = ln(x + sqr(x**2 + 1))
saida:
 parse source . calltype .
 if calltype = "COMMAND" then say y
 else return y
exit
```

INVERSE HYPERBOLIC COSINE

```
/*= REXX ============================================================*/
/*   ACOSH                                                           */
/*   Inverse Hyperbolic Cosin.  Answer is in degrees.                */
/*===================================================================*/
arg x .
if datatype(x,"N") = Ø | x < 1 then do
   y = "Error"
   signal saida
end
numeric digits 2Ø
y = ln(x + sqr(x**2 - 1))
y = trunc(y,17)
saida:
 parse source . calltype .
 if calltype = "COMMAND" then say y
 else return y
exit
```

INVERSE HYPERBOLIC COSINE

```
/*= REXX ============================================================*/
/*   ATANH                                                           */
/*   Inverse Hyperbolic Tangent.  Answer is in degrees.              */
/*===================================================================*/
arg x .
if datatype(x,"N") = Ø | abs(x) = 1 | abs(x) > 1 then do
   y = "Error"
   signal saida
end
numeric digits 2Ø
y = (ln(1 + x) - ln(1 - x))/2
y = trunc(y,17)
saida:
 parse source . calltype .
 if calltype = "COMMAND" then say y
 else return y
exit
```

A file trimming utility

Many processes or utilities produce information that is saved to a log or audit trail file. Quite often information is appended into this file to provide a historical perspective of the process. If left unchecked, these types of files will continue to grow until they exceed their size limits. We have developed a very simple utility that can be used to trim a file to keep the growth in check.

We opted to use REXX to develop the utility that we call FTRIMMER. Developing in REXX provided us with a very flexible programming environment enabling quick code generation and easy debugging. REXX has all the constructs of a structured environment with excellent I/O facilities, and a nice mechanism for trapping error conditions and allowing us to take the appropriate action. In the future we will present a similar utility that has been developed in Assembler code that accomplishes the same task, but is more efficient from a CPU consumption perspective.

FTRIMMER is very simple to invoke, requiring only a minimum of JCL to execute. Here is an example of what is required. You will need to make the appropriate modifications for your environment.

```
//myjob   JOB account,my name,CLASS=?,MSGCLASS=?
//REXXSTEP EXEC PGM=IKJEFT1A,REGION=ØM
//SYSTSPRT DD SYSOUT=*
//SYSEXEC  DD DISP=SHR,DSN=my.rexx.library
//SYSTSIN  DD *
%FTRIMMER DSN=my.file.to.trim TRIM=HEAD RETAIN=1ØØØ
```

Please note that you need to specify three parameters when you invoke the program. The DSN parameter provides the name of the dataset on which we want to perform the trim operation. The TRIM parameter indicates whether we are going to trim from the HEAD (or beginning of the file) or the TAIL (or end of the file). The RETAIN value specifies how many records we want to have remaining in the dataset after the trimming is complete. You can specify the parameters in any order. All of the invocation examples below will execute correctly.

```
%FTRIMMER DSN=my.file RETAIN=1ØØØ TRIM=TAIL
%FTRIMMER TRIM=HEAD DSN=my.file RETAIN=5ØØØ
%FTRIMMER RETAIN=1ØØØ TRIM=TAIL DSN=my.file
```

We have endeavoured to make the invocation as simple as possible. If you do not specify all three of the required parameters, the program will issue an error message and terminate.

Please note that we have utilized the EXECIO feature of REXX to perform all of the I/O on the dataset. If you take a look at the specific coding, you will see a line that looks like the following:

```
"EXECIO * DISKR TRIMIN (FINIS STEM trimin."
```

Let's examine the line a little more closely to understand what is occurring. THE EXECIO invokes the I/O facility, with DISKR specifying that we want to read the dataset. The asterisk indicates that we want to read the entire dataset that has been allocated to the TRIMIN DD name. The FINIS option calls for the file to be closed when it has been completely read. The STEM option indicates that we want all the records to be placed in the stem variable trimin. Depending on the size of the file, you may encounter some virtual storage issues attempting to read the entire file, although we have never had any difficulties. You may be able to simply change the REGION value on your job or step card to circumvent this. If you are not familiar with stem variables, they are a very flexible way of referencing data. When the EXECIO operation terminates, we can examine trimin.0 to see how many records were actually read in from the dataset. We now have a one-dimensional array, trimin.1, trimin.2, etc, which provides easy access and reference to the data. This indexing allows us to specify starting and ending values for our copy operation from trimin to the stem variable trimout.

The code is easily adaptable. You could add file back-up prior to the trim operation, or extended error messages. We have utilized a simple routine called EXSTATUS for displaying all our messages in a standardized format. EXSTATUS can be adapted to suit your needs as well.

FTRIMMER REXX EXEC

```
/* Rexx FTRIMMER                                                      */
/* ------------------------------------------------------------------+
 | FTRIMMER is a file trimming routine.  As designed it will work on |
 | physical sequential datasets.  The datasets can be either fixed   |
 | or variable, blocked or non-blocked.                              |
 |                                                                   |
 | PLIST   : DSN=name of the dataset to trim                         |
 |           TRIM=HEAD | TAIL indicates where trimming should occur  |
 |           RETAIN=number of lines to keep in the file              |
 |                                                                   |
 | CALLS   : EXSTATUS for uniform status messages                    |
 +------------------------------------------------------------------ */
pgmid='FTRIMMER'
myRC=Ø
/* ------------------------------------------------------------------+
 | Turn off messages and our prefix.                                 |
 +------------------------------------------------------------------ */
myRC=MSG('OFF')
"PROFILE NOPREFIX"
call exstatus pgmid,'start'
/* ------------------------------------------------------------------+
 | Establish routine to handle errors.                               |
 +------------------------------------------------------------------ */
call on error   name ExitPoint
/* ------------------------------------------------------------------+
 | Pull in the passed arguments.  We are expecting to get three      |
 | passed to us.                                                     |
 +------------------------------------------------------------------ */
arg arg1 arg2 arg3 .
/* -----------------------------------------------------------------+
 | Parse out the passed arguments and determine the keyword and      |
 | value for each of them.                                           |
 +------------------------------------------------------------------ */
parse var arg1 kw.1 '=' kval.1
parse var arg2 kw.2 '=' kval.2
parse var arg3 kw.3 '=' kval.3
/* ------------------------------------------------------------------+
 | Go through the keyword values and assign them to their respective |
 | variables.                                                        |
 +------------------------------------------------------------------ */
do iloop=1 to 3
 select
  when kw.iloop = 'DSN' then
   trimdsn=kval.iloop
  when kw.iloop = 'TRIM' then
   trimspot=kval.iloop
  when kw.iloop = 'RETAIN' then
   rvalue=kval.iloop
  otherwise
   do
    myRC=12
    mtxt='Unknown keyword specified, value =' kw.iloop
    call exstatus pgmid,'msgdi',mtxt
   end
 end
end
/* -------------------------------------------------------------------+
 | Validate the the TRIM value.  It should be either HEAD or TAIL    |
 +------------------------------------------------------------------- */
if trimspot ¬= 'HEAD' & trimspot ¬= 'TAIL' then
 myRC=12
/* -------------------------------------------------------------------+
 | Validate the RETAIN value.                                        |
 +------------------------------------------------------------------- */
dtype=datatype(rvalue,'N')
if dtype = 1 then
 rvalue=rvalue+Ø
else
 myRC=12
/* -------------------------------------------------------------------+
 | Validate the dataset.  Simple check to see whether the dataset is |
 | physical sequential and fixed or variable format.                 |
 +------------------------------------------------------------------- */
dsiRC=listdsi(trimdsn)
if dsiRC <= 4 then
 if left(sysdsorg,2) = 'PS' & ,
   (left(sysrecfm,1) = 'F' | left(sysrecfm,1) = 'V') then
  myRC=Ø
 else
  do
   mtxt=trimdsn 'not PS and Fixed or Variable format'
   call exstatus pgmid,'msgdi',mtxt
   myRC=12
  end
else
 do
  mtxt='Error performing listdsi function, RC=' dsiRC
  call exstatus pgmid,'msgdi',mtxt
  myrc=12;
 end
/* -------------------------------------------------------------------+
 | Check to see whether we had any errors with the keyword values.   |
 +------------------------------------------------------------------- */
if myRC ¬= Ø then signal ExitRtn
mtxt='Dataset to be trimmed is' trimdsn
call exstatus pgmid,'msgdi',mtxt
mtxt='Data will be trimmed from the' trimspot 'of the dataset'
call exstatus pgmid,'msgnb',mtxt
```

```
mtxt='Number of records retained in dataset will be' rvalue
call exstatus pgmid,'msgnb',mtxt
/* ------------------------------------------------------------------+
 | Allocate the dataset so we can read from it, perform the read, and|
 | then de-allocate the dataset. The stem variable trimin. will hold |
 | the records from the file.                                        |
 +------------------------------------------------------------------ */
mtxt='Allocating dataset' trimdsn
call exstatus pgmid,'msgdi',mtxt
"ALLOC F(TRIMIN) DS("trimdsn") SHR"
if RC ¬= Ø then
 do
  myRC = RC
  mtxt='Error encountered performing allocate. RC=' myRC
  call exstatus pgmid,'msgdi',mtxt
  signal ExitRtn
 end
"EXECIO * DISKR TRIMIN (FINIS STEM trimin."
if RC ¬= Ø then
 do
  myRC = RC
  mtxt='Error encountered performing EXECIO. RC=' myRC
  call exstatus pgmid,'msgdi',mtxt
  signal ExitRtn
 end
/* ------------------------------------------------------------------+
 | EXECIO completed with no errors.  Echo out the record count and   |
 | free the dataset.                                                 |
 +------------------------------------------------------------------ */
mtxt=trimin.Ø 'records read from' trimdsn
call exstatus pgmid,'msgnb',mtxt
"FREE F(TRIMIN)"
if RC ¬= Ø then
 do
  myRC = RC
  mtxt='Error encountered performing deallocate. RC=' myRC
  call exstatus pgmid,'msgdi',mtxt
  signal ExitRtn
 end
mtxt='Dataset' trimdsn 'has been de-allocated'
call exstatus pgmid,'msgnb',mtxt
/* ------------------------------------------------------------------+
 | Determine whether we need to copy records, and, if we do, we will |
 | prime the variables startval and endval with the correct values so|
 | that we can use a simple loop to copy data from stem variable     |
 | trimin to trimout.                                                |
 +------------------------------------------------------------------ */
if trimin.Ø <= rvalue then
 do
  mtxt='Number of records in dataset <= to RETAIN value of' rvalue
  call exstatus pgmid,'msgdi',mtxt
  mtxt='No trimming will be performed'
  call exstatus pgmid,'msgnb',mtxt
  signal ExitRtn
 end
if trimspot = 'TAIL' then
 do
  startval=1
  endval=rvalue
 end
else
 do
  startval=(trimin.Ø-rvalue)+1
  endval=trimin.Ø
 end
/* ------------------------------------------------------------------+
 | Loop control variables startval and endval have been primed, copy |
 | the records.                                                      |
 +------------------------------------------------------------------ */
jloop=Ø
do iloop = startval to endval
 jloop=jloop+1
 trimout.jloop=trimin.iloop
end
trimout.Ø=jloop
/* ------------------------------------------------------------------+
 | Allocate the dataset so we can write to it, perform the write and |
 | then de-allocate the dataset. The stem variable trimout. has the  |
 | the records we will write to the file.                            |
 +------------------------------------------------------------------ */
mtxt='Allocating dataset' trimdsn
call exstatus pgmid,'msgdi',mtxt
"ALLOC F(TRIMOUT) DS("trimdsn") OLD"
if RC ¬= Ø then
 do
  myRC = RC
  mtxt='Error encountered performing allocate. RC=' myRC
  call exstatus pgmid,'msgdi',mtxt
  signal ExitRtn
 end
"EXECIO * DISKW TRIMOUT (FINIS STEM trimout."
if RC ¬= Ø then
 do
  myRC = RC
  mtxt='Error encountered performing EXECIO. RC=' myRC
  call exstatus pgmid,'msgdi',mtxt
  signal ExitRtn
 end
mtxt=trimout.Ø 'records written to' trimdsn
call exstatus pgmid,'msgnb',mtxt
```

```
"FREE F(TRIMOUT)"
if RC ¬= Ø then
 do
  myRC = RC
  mtxt='Error encountered performing deallocate. RC=' myRC
  call exstatus pgmid,'msgdi',mtxt
  signal ExitRtn
 end
mtxt='Dataset' trimdsn 'has been de-allocated'
call exstatus pgmid,'msgnb',mtxt
/* ------------------------------------------------------------------+
 | Single exit point for normal termination of the routine.        |
 +------------------------------------------------------------------ */
ExitRtn:
call exstatus pgmid,'stopn'
exit myRC
/* ------------------------------------------------------------------+
 | The following routine is used to trap errors and make a decision  |
 | whether we can proceed, or halt the execution with a non-zero     |
 | return code. We are using a variable myRCadj to help formulate    |
 | the return code we will present, and to assist with diagnostics.  |
 +------------------------------------------------------------------ */
ExitPoint:
myRC=RC
etype = condition('C')
edesc = condition('D')
einst = condition('I')
say 'Call on' etype 'invoked; RC=' RC ' ; instruction=' einst
say 'at line' sigl ' , Desc=' edesc
call exstatus pgmid,'stopa'
Exit myRC
```

EXSTATUS REXX EXEC

```
/* Rexx EXSTATUS                                                     */
/* ------------------------------------------------------------------+
 | Rexx EXSTATUS is used to place messages out in the SYSTSPRT       |
 | output dataset.  By using the routine, we can enforce and pro-    |
 | vide a common message format.                                     |
 |                                                                   |
 | PARMS   : Routine is invoked with three arguments                 |
 |                                                                   |
 | arg1    : Name of the calling REXX routine                        |
 | arg2    : Type of message.  Possible values are:                  |
 |           start - message indicating routine is beginning         |
 |           stopn - message indicating routine is ending normally   |
 |           stopa - message indicating routine is ending abnormally |
 |           msgdi - routine is requesting display of current msg    |
 |                   with a blank line prior to the message          |
 |           msgnb - routine is requesting display of current msg    |
 |                   without the additional blank line               |
 | arg3    : Message text for MSGDI request                          |
 +------------------------------------------------------------------ */
/* -----------------------------------------------------------------+
 | Pick up our passed arguments.  We will always get arg1 and arg2.  |
 | We will have a value in arg3 when the function is MSGDI.          |
 +------------------------------------------------------------------ */
arg1=arg(1)
arg2=arg(2)
arg3=arg(3)
/* -----------------------------------------------------------------+
 | Assign the passed arguments into local variables.  We also want   |
 | to translate arg1 and arg2 to upper case.                         |
 +------------------------------------------------------------------ */
parse upper var arg1 cpgm
parse upper var arg2 mtype
parse var arg3 msgtxt
outext=' '
/* -----------------------------------------------------------------+
 | Populate outext based on the function call.                       |
 +------------------------------------------------------------------ */
if mtype='START' then
  outext='- Beginning program execution'
if mtype='STOPN' then
  outext='- Program is terminating normally'
if mtype='STOPA' then
  outext='- Program is terminating abnormally'
if mtype='MSGDI' | mtype='MSGNB' then
  outext='-' msgtxt
/* -----------------------------------------------------------------+
 | If it was a known function, outext will have a value.             |
 +------------------------------------------------------------------ */
if outext¬=' ' then
  if mtype='MSGNB' then
    say date('U') time('N') cpgm outext
  else
    do
      say ' '
      say date('U') time('N') cpgm outext
    end
/* -----------------------------------------------------------------+
 | Single exit point from the routine.                               |
 +------------------------------------------------------------------ */
ExitRtn:
return
```

William Meany
Enterprise Data Technologies (USA)

Job return code monitoring

THE PROBLEM

There will always be a need in production environments to be notified in a timely manner if any production job fails. Though there might be an operator to monitor things, it is nice to be notified by the system as soon as possible. What happens if the operator fails to notice? If there is an automated way to monitor and notify the operator with an alert message on the console and send an e-mail or page to the appropriate production support person, the corrective action can be taken at the earliest possible opportunity. Especially in a mission-critical production environment, it is mandatory to monitor all the production batch jobs and system-related jobs such as daily, weekly, monthly back-ups and SMF jobs. If a particular batch job fails, it is most important to notify the correct person.

A SOLUTION

The following CLIST program, which uses IOF Version 7d or higher, solves this problem. This program reads three input files – RCCHECK.INCLUDE, RCCHECK.EXCLUDE, and RCCHECK.CALENDAR. RCCHECK.INCLUDE contains the most important jobs to be monitored along with what kind of notification we need. RCCHECK.EXCLUDE contains the same production jobs along with step names for return codes to ignore: for example, if a job always finishes STEP1 with return code 4, which is normal, we do not need any notification. RCCHECK.CALENDAR contains information about the dates on which particular people are supposed to get information. This will be useful when there is a rota for responsibility in production support.

Three kinds of notification can be achieved with this program if any production job fails. One is a highlighted attention console message to the operator. A message should be sent to the console, regarding who should be contacted for a particular production job failure. There should be a written document containing detailed action to be taken. In order to achieve this, the CLIST program will call an Assembler routine (WTOCLIST) and pass a message such as ‘Production JOB PRODBKUP failed with RC 12 – Please call Production Support’. The Assembler routine will put the message on the operator console, highlighted, to grab the operator’s attention

The second notification can be a pager message to the production support team, if your environment supports that facility. These instructions are commented out in the CLIST program, which can be modified as per your environment.

The other method of notification can be an e-mail notification to the person concerned.

This routine is designed to be run in TSO batch every 30 minutes. It goes into IOF, lists the requested jobs that have been completed in the last hour, reviews their return codes for each step, and then, if there is a non-zero return code (which is not excluded) it beeps (pages) the appropriate person. It uses the IOFPAGE1 routine, passing it the list item (job) about which to get the error information and the pager name to be paged. The IOFPAGE1 routine then checks the exclusion file RCCHECK.EXCLUDE and, if the job passed is not in the exclusion file, it pages the requested pager. This process is accomplished for all jobs that receive a non-zero return code. This routine accepts two parameters. One is ALERTMSG, which indicates which alert mechanism is to be used. Valid values are PAGE (the default), EMAIL, or BOTH. PAGE causes a page (beeper page) to happen, EMAIL sends an e-mail, and BOTH initiates both a page and an e-mail. If not specified, it defaults to paging and/or e-mailing, based on whether the include file has a non-blank entry for the page and e-mail fields.

The second parameter is INCLUDEFILE, which identifies the sequential file containing the job names to look for. The default include file is RCCHECK.INCLUDE if not specified. The format of the file must be JOBNAME in columns 1–8 (an * may be used at the end of the jobname to indicate all jobnames that start with the characters before the *). Columns 10–18 contain the pagerid of the person to be beeped and columns 20–58 contain the e-mail ID of the person to be e-mailed. If the word ‘CALENDAR’ is in columns 60-67, a filename should be in columns 70–115 for a calendar file.

The calendar file format has the following layout. Dates are shown in columns 1–8, the pagerid for a pager in columns 10–18, and e-mail ID in columns 20–58 as in the INDCLUDEFILE. The format of the date in the calendar file is YYMMDD with the records in ascending chronological order.

This REXX routine can be run in batch. Sample JCL is also provided. Whenever there is a message or page from this routine, it will record an entry with the date and time in the log dataset. This is just for future reference to find out which production job has failed or is not up at specific time.

With the help of JES2 automatic commands, you can run this REXX routine in batch every 30 or 60 minutes, depending on your needs. A sample JES2 automatic command to run every 15 minutes would be:

```
$TAAØØ1,I=3ØØ,'$VS,''S CHECKACT'''
```

OPERATIONAL ENVIRONMENT

Use of this program is dependent on the correct customization of IOF and the Assembler routine must have been compiled and link edited in dataset CHECK.ACTIVE.LOADLIB. The sample JCL procedure has to be in any one of the JES2 procedure libraries with the proper user-id, which has privileges to access the input and log datasets. IOF minimum release should be 7D.

RCCHECK – CLIST PROGRAM

```
/* - - RCCHECK- - Check return codes of jobs in a list, and notify    */
/*     at console or send page/email to the concerned person          */
/*                                                                    */
PROC Ø ALERTMSG(PAGE) INCLUDEFILE('RCCHECK.INCLUDE')
```

```
CONTROL END(ENDO) NOLIST NOCONLIST NOFLUSH ASIS
                   /*                                              */
SET ERRORS  = 6          /*Maximum errors before termination     */
SET NOJOBS = &STR(NO)    /*Assume there will be jobs in the list */
SET MAXERRS = &ERRORS
SET ENV = IOFDOWN        /* IOF IS NOT ACTIVE IN THIS SESSION YET */
SET ALERTSND = &STR(NO)  /* NO JOBS HAVE QUALIFIED FOR ALERT YET  */
/*  Initialize our error routine. Error numbers less than 256 are    */
/*  allowed to continue, because they are actually return codes      */
/*  rather than errors. A few real errors (6) are allowed before     */
/*  the CLIST is terminated. The IOF long error message is displayed */
/*  for all real errors.                                             */
SET ERRORCT = Ø
ERROR DO
  SET RC = &LASTCC
  IF &RC EQ 14Ø4 THEN DO           /* No jobs in list - no match */
     SET NOJOBS = &STR(YES)
     RETURN CODE(&RC)
     ENDO
  IF &RC EQ 15Ø3 THEN DO           /* No jobs in list - security */
     SET NOJOBS = &STR(YES)
     WRITE NO JOBS FOUND IN JES2 BY THIS NAME
     RETURN CODE(&RC)
     ENDO
  IF &RC LT 256 OR &RC EQ 13Ø7 THEN RETURN CODE(&RC)
  IF &RC GE 99Ø AND &RC LE 999 THEN RETURN CODE(&RC)
  SET ERRORCT = &ERRORCT + 1
  SET MESSAGE = Error
  IF &ENV EQ IOFUP THEN +
    TSICOPY NAME(MESSAGE) SECTION(PANEL) TO(CLIST)
  IF &RC NE 4ØØØ THEN +
    WRITE CHECKRC  &SYSDATE &SYSTIME Error &RC
  WRITE CHECKRC  &SYSDATE &SYSTIME &MESSAGE
  IF &RC = 311 THEN DO
    WRITE CHECKRC ERROR - AUTHORIZED VAR(&ASCBTYPE) VAR2(&GRPNAME)
    JUMP X
    EXIT CODE(16)
    ENDO
  IF &RC = 3ØØ14 OR &RC = 3ØØ15 THEN DO
    WRITE CHECKRC ERROR - LICSENCE EXPIRED FOR IOF PRODUCT
    JUMP X
    EXIT CODE(16)
    ENDO
  IF &ERRORCT GT &MAXERRS THEN DO
    WRITE CHECKRC  &SYSDATE &SYSTIME Maximum error count exceeded, +
          Terminating.
    EXIT CODE(16)
    ENDO
  RETURN CODE(&RC)
  ENDO /*ERROR ROUTINE*/
GLOBAL IN1 CHCKJN CHCKUID CHCKPID CALENDAR FNAME PAGECNT FIRSTTM
GLOBAL CHCKUID1 CHCKPID1 CHCKUID2 CHCKPID2 CHCKUID3 CHCKPID3
GLOBAL CHCKUID4 CHCKPID4 CHCKUID5 CHCKPID5 CHCKUID6 CHCKPID6
GLOBAL CHCKUID7 CHCKPID7 CHCKUID8 CHCKPID8 CHCKUID9 CHCKPID9
SET FIRSTTM = 'NO'
SET RCX = Ø
DO WHILE &RCX NE &STR(FOREVER) /* loop through all of include file*/
SET PAGECNT = Ø                /* Start with zero people to page  */
SET CALENDAR = &STR(NO)        /* Assume no calendar file         */
SET NOJOBS = &STR(NO)          /* Assume there WILL BE jobs in list*/
SYSCALL CHECKJOB
SET RCX = &LASTCC
IF &RCX EQ 98 THEN GOTO GETOUT  /*NORMAL END OF FILE */
IF &RCX EQ 99 THEN GOTO GETOUT  /*NORMAL END OF FILE */
IF &CALENDAR = &STR(YES) THEN SYSCALL CHECKCAL
SET RCX = &LASTCC
/*  Get a list of all the jobs listed in the include file record    */
SET CHCKJN2 = &CHCKJN&STR(*)
IOF * JOBNAME(&STR(&CHCKJN2)) OUTPUT CLIST
SET &IOFRC = &LASTCC
IF &IOFRC = Ø THEN SET ENV = IOFUP
WRITE CHECKING JOBS(&CHCKJN2) RC=&IOFRC FROM INVOKING IOF
IF &NOJOBS EQ &STR(YES) THEN GOTO NEXTJOB /*There are no jobs in lst*/
EXTEND
TSICOPY NAME(INPNODE) TO(CLIST)
SET MYNODE = &INPNODE            /*JES NODE ID - possible future use*/
LOCK
/*  Exclude jobs that do not start with P76P*                       */
EXCLUDE JOBNAME NB &CHCKJN
IF &NOJOBS EQ &STR(YES) THEN GOTO NEXTJOB /*There are no jobs in lst*/
/*  Exclude jobs not run in the last hour                           */
/*  Get the current date and time and then calculate one hour ago   */
SET DDATE = &STR(&SYSDATE)
SET DMM = &SUBSTR(1:2,&DDATE)      /* month    */
SET DMM = &DMM                     /* Convert from string to number*/
SET DDD = &SUBSTR(4:5,&DDATE)      /* day      */
SET DYY = &SUBSTR(7:8,&DDATE)      /* year     */
SET TTIME = &STR(&SYSTIME)
SET THH = &SUBSTR(1:2,&TTIME)      /* hour     */
SET CHH = &SUBSTR(1:2,&TTIME)      /* Current hour */
SET TMM = &SUBSTR(4:5,&TTIME)      /* minute   */
SET TSS = &SUBSTR(7:8,&TTIME)      /* second   */
SET CTIME = &STR(&CHH)&STR(:)&STR(&TMM) /*current time */
SET JDATE = &SYSJDATE
SET DAY = &SUBSTR(4:6,&JDATE)
SET CDAY = &SUBSTR(4:6,&JDATE)      /* current day */
IF &THH = &STR(ØØ) THEN DO         /*Handle the hour and day boundary*/
   SET THH = &STR(24)              /* Handle the hour/day boundary   */
   SET DDD = &DDD - 1              /* Handle the day boundary        */
   SET DAY = &DAY - 1              /* Handle the day boundary        */
   IF &DDD = Ø THEN DO      /*Handle month and year boundary*/
      SET DDD = &STR(31) /*Handle most months boundary*/
      IF &DMM = 3 THEN SET DDD = &STR(28) /*Handle month boundary*/
      IF &DMM = 5 THEN SET DDD = &STR(3Ø) /*Handle month boundary*/
      IF &DMM = 7 THEN SET DDD = &STR(3Ø) /*Handle month boundary*/
      IF &DMM = 8 THEN SET DDD = &STR(3Ø) /*Handle month boundary*/
      IF &DMM = 1Ø THEN SET DDD = &STR(3Ø) /*Handle month boundary*/
      IF &DMM = 12 THEN SET DDD = &STR(3Ø) /*Handle month boundary*/
      SET DMM = &DMM - 1                  /*Move back one month */
      IF &DMM = Ø THEN SET DMM = &STR(12) /*Handle year boundary */
      ENDO
   ENDO
SET &THH = &THH - 1                /* Set the hour back by one */
IF &LENGTH(&STR(&THH)) = 1 THEN SET THH = &STR( )&THH
IF &LENGTH(&STR(&DMM)) = 1 THEN SET DMM = &STR(Ø)&DMM
IF &LENGTH(&STR(&DDD)) = 1 THEN SET DDD = &STR(Ø)&DDD
IF &LENGTH(&STR(&DAY)) = 1 THEN SET DAY = &STR(Ø)&DAY
IF &LENGTH(&STR(&DAY)) = 2 THEN SET DAY = &STR(Ø)&DAY
SET FTIME = &STR(&THH)&STR(:)&STR(&TMM)
SET FDATE = &STR(&DYY)&STR(&DMM)&STR(&DDD)
SET CMPRDT = &FDATE &FTIME
SET JDATEFORMAT = &DAY &FTIME
SET CDATEFORMAT = &CDAY &CTIME          /* current day and time */
WRITE FTIME=&FTIME CMPRDT=&CMPRDT DMM=&DMM DDD=&DDD C=&CDATEFORMAT
WRITE J=&JDATEFORMAT
/* Exclude other IOF sections but keep the OUTPUT section */
EXCLUDE SECTION NE OUTPUT
IF &NOJOBS EQ &STR(YES) THEN GOTO NEXTJOB /*There are no jobs in lst*/
/* Exclude jobs that did not complete in the last 6Ø minutes       */
/* EXCLUDE DATETIME LT &STR('&CMPRDT') */
EXCLUDE TIMECOND GT &STR('&CDATEFORMAT')
EXCLUDE TIMECOND LT &STR('&JDATEFORMAT')
IF &NOJOBS EQ &STR(YES) THEN GOTO NEXTJOB /*There are no jobs in lst*/
/*  Do the RC command to find bad return code jobs only            */
RC Nozero
/*  Loop through all jobs on the menu. Beep or e-mail each person   */
/*  as listed in either the include or calender file.              */
CONTROL END(ENDO)
UP MAX
TSICOPY NAME(ROWS) SECTION(PANEL) TO(CLIST)
IF &ROWS = Ø THEN GOTO NEXTJOB          /*There are no jobs in list*/
SET JOBLNUM = Ø
SET LASTCC = Ø
DO WHILE &LASTCC LT 8         /*Loop through all jobs in list   */
SET JOBLNUM = &JOBLNUM + 1
  TSICOPY NAME(JOBNAME JOBID ) TO(CLIST)
  DO &I=1 TO &PAGECNT BY 1    /*Loop through all people to Alert*/
```

```
    SET MID = &&CHCKUID&I
    SET PID = &&CHCKPID&I
    SET MID = &MID
    SET PID = &PID
    WRITE IOFPAGE1 &JOBNAME &JOBID PAGE(&PID) MAIL(&MID) CNT(&PAGECNT)
    IF &PID NE THEN DO
       SYSCALL IOFPAGE1 &PID &JOBLNUM
       SET IOFPRC = &LASTCC       /*Remember return code */
       IF &IOFPRC = Ø THEN SET ALERTSND = &STR(YES)
       ENDO /*IF*/
    ENDO /*&I=1*/
  DOWNCTRL
  ENDO /*DO WHILE &LASTCC*/    /*End processing all jobs*/
SET RCX = Ø
NEXTJOB: JUMP X /*IOF command */
SET ENV = IOFDOWN        /* IOF IS NOT ACTIVE IN THIS SESSION NOW */
ENDO /*DO WHILE -FOR EACH RECORD IN THE INCLUDE FILE */
GETOUT: SET BLAH = Ø
IF &ALERTSND NE &STR(YES) THEN WRITE NO ALERTS SENT - NO JOBS QUALIFY
IF &RCX = 99 THEN EXIT CODE(99)
EXIT CODE(Ø)
END /*PROC CHECKRC*/
CHECKJOB: PROC Ø
/*This routine checks the include file to see which job names    */
/*should be included in the return code checking process. It just*/
/* goes line by line through the inclusion file, filing out the  */
/* three global variables (CHCKJN CHCKUID CHCKPID) for the    */
/* JOBNAME, userid (for email) and the pager id.                 */
CONTROL NOFLUSH NOMSG END(ENDO)
/* GET THE JOBNAMES TO CHECK THE RETURN CODES OF FROM INCLUSION FILE*/
SET RCX = Ø
IF &FIRSTTM = &STR(YES) THEN GOTO GETREC
SET FIRSTTM = &STR(YES)    /* only need to open file once*/
ALLOCATE FILE(IN1) DA('RCCHECK.INCLUDE') SHR
OPENFILE IN1 INPUT
/* IOFNEST */
/* Now check the field against each jobname entry in the file*/
GETREC: GETFILE IN1
SET RCX = &LASTCC
IF &SUBSTR(1:8,&IN1) = &STR(EOF.....) THEN DO
   CLOSFILE IN1
   FREE FILE(IN1)
   RETURN CODE(98)
   ENDO
IF &RCX NE Ø THEN DO
   WRITE LAST GETFILE ON INCLUSION FILE - RC(&RCX)
   CLOSFILE IN1
   TSO FREE FILE(IN1)
   RETURN CODE(99)
   ENDO
IF &SUBSTR(1:1,&IN1) = &STR(*) THEN GOTO GETREC /* Allow comments*/
/*Get the jobname, userid(for email), and pagerid from include file*/
SET PAGECNT = &PAGECNT + 1
SET CHCKJN1 = &SUBSTR(1:8,&IN1)
SET CHCKPID = &SUBSTR(1Ø:18,&IN1)
SET CHCKPID&PAGECNT = &CHCKPID       /* Default Pager id    */
SET CHCKUID = &SUBSTR(2Ø:58,&IN1)
SET CHCKUID&PAGECNT = &CHCKUID       /* Default mail id     */
/* Check and see if we need to look up person to page from a calendar*/
SET CFIELD  = &SUBSTR(6Ø:67,&IN1)
IF &CFIELD  EQ &STR(CALENDAR) THEN DO
   SET CALENDAR = &STR(YES)
   SET FNAME = &SUBSTR(7Ø:115,&IN1)  /* Calendar file dset name */
   SET FNAME = &FNAME
   ENDO
SET CHCKJN = &STR()
SET JCNTR = 1
DO WHILE &JCNTR LE 8
   SET CHCKCH = &SUBSTR(&JCNTR:&JCNTR,&CHCKJN1)
   IF &STR(&CHCKCH) NE &STR(*) THEN +
      IF &STR(&CHCKCH) NE &STR( ) THEN +
         SET CHCKJN = &STR(&CHCKJN)&STR(&CHCKCH)
   SET JCNTR = &JCNTR + 1
   ENDO
/*WRITE CHCKJN=&CHCKJN &CHCKUID &CHCKUID1 &CHCKPID &CHCKPID1 */
RETURN CODE(Ø)
ENDO /*PROC CHECKJOB*/
CHECKCAL: PROC Ø
/*This routine checks the Calendar file to find who to page or    */
/*e-mail based on the day of the year. It just reads each line in */
/*the file until it finds a match, then it gets the info from the */
/* file. global variables (CHCKUID CHCKPID) are for the userid    */
/* (for email) and the pager id.                                  */
CONTROL NOFLUSH NOMSG END(ENDO)
/* GET THE JOBNAMES TO CHECK THE RETURN CODES OF FROM INCLUSION FILE*/
ALLOCATE FILE(IN2) DA('&FNAME') SHR
OPENFILE IN2 INPUT
/* Get todays date */
SET DDATE = &STR(&SYSDATE)
SET DMM = &SUBSTR(1:2,&DDATE)      /* month    */
SET DDD = &SUBSTR(4:5,&DDATE)      /* day      */
SET DYY = &SUBSTR(7:8,&DDATE)      /* year     */
SET FDATE = &DYY&DMM&DDD
/* Now check the first field and compare todays date*/
SET PAGECNT = Ø
SET RCX = Ø
SET RCY = Ø
DO WHILE &RCX NE 98
```

```
   DO WHILE &RCX EQ Ø
GETNEXT: GETFILE IN2
      SET RCX = &LASTCC
      IF &SUBSTR(1:8,&IN2) = &STR(EOF.....) THEN SET RCX = 98
      IF &SUBSTR(1:1,&IN2) = &STR(*) THEN GOTO GETNEXT /*Comments OK*/
      SET CHCKDAT = &SUBSTR(1:6,&IN2)
      IF &CHCKDAT = &FDATE THEN SET RCY = 99 /*Indicate the match*/
      /*If we had a match and now no longer a match time to end */
      IF &CHCKDAT NE &FDATE THEN IF &RCY = 99 THEN SET RCX = 98
      IF &CHCKDAT NE &FDATE THEN IF &RCY = 99 THEN SET RCY = Ø
   /*Get the userid(for email), and pagerid from file    */
   IF &RCY = 99 THEN DO  /* This is a match in the calendar file */
      SET PAGECNT = &PAGECNT + 1        /* Count users to page   */
      SET CHCKPID = &SUBSTR(1Ø:18,&IN2) /* Get pager id          */
      SET CHCKPID&PAGECNT = &CHCKPID    /* Strip blanks          */
      SET CHCKUID = &SUBSTR(2Ø:58,&IN2) /* Get users mailid      */
      SET CHCKUID&PAGECNT = &CHCKUID    /* Strip blanks          */
   /* WRITE CHCKDAT=&CHCKDAT CHCKUID=&CHCKUID CHCKPID=&CHCKPID */
      ENDO /*IF RCY EQ 99 */
   ENDO /*WHILE RCX EQ Ø */
ENDO /* WHILE RCX NE 98 */
/*IF THERE WERE NO ENTRIES IN THE DATE FILE USE THE DEFAULT ENTRIES*/
/*FROM THE INCLUDE FILE FOR MAILID AND PAGEID. THEY WERE SET BEFORE*/
/*THE CALENDAR ROUTINE WAS CALLED. Next line does this.            */
IF &PAGECNT = Ø THEN SET PAGECNT = 1
CLOSFILE IN2
TSO FREE FILE(IN2)
RETURN CODE(&RCX)
ENDO /*PROC CHECKCAL*/
IOFPAGE1: PROC 2 PAGER JLISTNO
/*CONTROL NOCAPS LIST MSG*/
CONTROL NOCAPS NOMSG NOFLUSH END(ENDO)
/*Get JOBNAME etc from the display screen panel*/
TSICOPY NAME(JOBNAME JOBID COMMENTS) SECTION(OUTPUT) TO(CLIST) +
ROW(&JLISTNO)
/*Select the job to get more detailed information*/
&JLISTNO SELECT
SET &STEP=&SUBSTR(24:31,&COMMENTS)
SET &STEP1=&STEP
SET &RCC=&SUBSTR(12:16,&COMMENTS)
SET &RCC  =&RCC
/* Search through all the steps to get the detailed information */
/* on the step with the highest return code.                    */
EXCL STEP NE &STEP1
SET RCX = Ø
DO UNTIL &RCX ¬= Ø
FIND STEP &STEP1
TSICOPY NAME(RC PGM STEP PRSTEP PROC) TO(CLIST)
SET RCX = &LASTCC
SET STEP = &STEP
SET RC   = &RC
IF &STEP1 = &STEP THEN DO
   IF &RC = &RCC THEN SET RCX = 1Ø
   IF &RC ¬= &RCC THEN DOWNCTRL
   ENDO ELSE DOWNCTRL
ENDO
/* END the IOF environment to return to the original display screen*/
END
/* Set up all the variables needed to search the exclusion file*/
GLOBAL JRC JRC2 JSTEP JPSTEP JPROC JCALL JLISTN2 JPGM JPRSTEP
SET JRC2 = &STR(&RC)
SET JSTEP = &STR(&STEP)
SET JPROC = &STR(&PROC)
SET JPGM  = &STR(&PGM)
SET JPRSTEP = &STR(&PRSTEP)
SET &UN =_
/* Get the date and time from the comments field made by RC command*/
SET &COMMENT =&SUBSTR(1:2,&COMMENTS)&UN&SUBSTR(4:5,&COMMENTS)
SET &COMMENT=&COMMENT&UN
SET &COLON=:
SET &HOUR=&SUBSTR(7:8,&COMMENTS)
SET &HOUR=&HOUR                     /*eliminate blanks*/
IF &LENGTH(&HOUR) = Ø THEN SET HOUR = &STR(ØØ)
IF &LENGTH(&HOUR) = 1 THEN SET HOUR = &STR(Ø)&HOUR
SET &COMMENT=&COMMENT&HOUR&COLON
SET &COMMENT=&COMMENT&SUBSTR(1Ø:11,&COMMENTS)&UN
SET &JRC    =&STR(&JRC2)
IF &SUBSTR(18:23,&COMMENTS) EQ &STR(serr) THEN SET JRC = &STR(SYS_ERR)
IF &JRC = Ø THEN DO
   WRITE HIGHEST RETURN CODE FOR JOB IS Ø - NO PAGING REQUIRED
   GOTO PAGEEXIT     /*Bail out if highest RC = Ø */
   ENDO
IF &SUBSTR(18:23,&COMMENTS) EQ &STR(serr) THEN SET JRC2 = &STR(SYS_ERR)
SET &ABND   =&JRC2
IF  &ABND EQ &STR(run) THEN SET &ABND = &STR(JCL_ERROR)
SET &ABND   =&STR(RC_)&ABND
SET &ABNDCODE=&SUBSTR(18:23,&COMMENTS)
SET &ABNDCODE=&ABNDCODE
IF &ABNDCODE EQ &STR(run) THEN SET ABND = &STR(JCL)
IF &ABNDCODE EQ &STR(run) THEN SET ABNDCODE = &STR(ERROR)
/* if jcl error set RC equal to the literal JCL */
IF &ABNDCODE EQ &STR(ERROR) THEN SET JRC = &STR(JCL)
SET &COMMENT=&COMMENT&UN&ABND&UN
IF &ABNDCODE ¬= THEN SET &COMMENT=&COMMENT&UN&ABNDCODE&UN
SET &STEPIT =STEPNAME_
SET &STEPNAME = &STR(&STEP)
SET &COMMENT = &COMMENT&STEPIT&STEPNAME
SET &JOBNAM1=&JOBNAME
SET &JOBIT =JOB_
SET &MSG1 = &JOBIT&JOBNAM1&UN&JOBID&UN&COMMENT
SET &PGMS =_PROGRAM_
SET &MSG1 = &MSG1&PGMS&PGM
/* IOFNEST *  */
/* EXTEND      */
/* NESTEXIT    */
/* Here is where we check and see if this jobstep is in the exclusion*/
/* file 'RCCHECK.exclude'
SYSCALL CHECKJB2 &JOBNAME
SET &CHECKRC = &LASTCC
/* WRITE CHECKJB2 LASTCC=&LASTCC */
IF &CHECKRC = Ø THEN DO
   WRITE JOB &JOBNAME &JOBID STEP &STEP RC(&JRC) IS IN +
         THE PAGE EXCLUSION FILE
   WRITE This jobs return code can be ignored. No page is necessary.
   RETURN CODE(99)
   ENDO
CONTROL CONLIST LIST
WRITE PAGING &PAGER MSG=&MSG1
/* This same routine can be used to send pager message to the        */
/* concerned Production programmer. Do uncomment the following       */
/* three lines to send page from the server where you run your paging*/
/* software. In our case, we run the paging software (TELALERT) on   */
/* one of our Unix systems. We use the REXEC command to send a page  */
/* from the mainframe.                                               */
/*        host = 'HOSTNAME'                                          */
/*       pager = 'PAGERID'                                           */
/*"REXEC -l loginid -p password "HOST" telalertc -g "pager" -m "msg1 */
/*                                                                   */
/*          CALL THE WTOREXX ASSEMBLER ROUTINE TO GET HIGHLIGHTED    */
/*          MESSAGE ON CONSOLE                                       */
            CALL WTOCLIST(msg1)SET &RCY = &LASTCC
WRITE RC FROM RXEC WAS &RCY
CONTROL NOCONLIST NOLIST
IF &RCY NE Ø THEN WRITE PAGE FAILED - TRY AGAIN
IF &RCY EQ Ø THEN DO
   WRITE PAGE REQUEST SENT
TSO SEND '* * * * * *O*P*E*R*A*T*O*R* * * * * * * * * * * * * * * *'
TSO SEND 'The following message was sent to pager====> &PAGER      '
TSO SEND '&MSG1'
TSO SEND '* * * * * * * * * * * * * * * * * * * * * * * * * * * * *'
   TSO ALLOCATE FILE(OUTPAGE) DA('RCCHECK.LOG') MOD
   OPENFILE OUTPAGE OUTPUT
   SET OUTPAGE = &STR(&SYSDATE)&UN&SYSTIME&UN&PAGER&UN&MSG1
   PUTFILE OUTPAGE
   CLOSFILE OUTPAGE
   TSO FREE FILE(OUTPAGE)
   ENDO
```

```
PAGEEXIT: RETURN CODE(Ø)
ENDO /* IOFPAGE1 PROC */
CHECKJB2: PROC 1 JOBNAME
/*This routine checks the exclusion file to see whether this job */
/*is to be excluded from being paged.                            */
/* It takes as input the jobname and outputs a return code Ø if  */
/* the job is NOT FOUND in the exclusion list, allowing the page */
/* to continue. It first checks the file for the jobname. If the */
/* jobname matches one in the file it then checks the other fields*/
/* to see whether they all match. If so it sets a return code of Ø   */
/* to indicate that this job jobstep proc procstep and job rc should */
/* not cause a page to happen.                                       */
CONTROL NOFLUSH NOMSG END(ENDO)
/* IOFNEST */
TSO ALLOCATE FILE(IN3) DA('RCCHECK.EXCLUDE') SHR
OPENFILE IN3 INPUT
/*CHECK IF JOBNAME IS IN THE FILE */
GLOBAL IN3 FNDFLG
/* Now check the field against each jobname entry in the file*/
CHECKJ2: SET &FNDFLG = NO
SET RCX = Ø
SET CHKRC4 = 99
DO WHILE &RCX = Ø
   GETFILE IN3
   SET RCX = &LASTCC
   /*Compare the first field in the file against the job's jobname*/
   SET &JOBNAME = &JOBNAME&STR(       )
   /* WRITE JOBNAME=&JOBNAME */
   SYSCALL CHKFLD 1 &SUBSTR(1:8,&JOBNAME)
   IF &LASTCC = Ø THEN SET &RCX = 99
   /*WRITE CHKFLD LASTCC WAS &RCX */
   IF &RCX ¬= 99 THEN SET &RCX = Ø
   /*WRITE CHKFLD LASTCC WAS &RCX after change */
   IF &SUBSTR(1:8,&IN3) = &STR(EOF.....) THEN SET RCX=EOF
   /*WRITE JOBNAME=&SUBSTR(1:8,&IN3) RCX=&RCX */
   ENDO
/*If RCX = 99 then there was a match on the jobname in the file     */
/*Here is where we check the other fields to verify a complete match*/
IF &RCX = 99 THEN DO
   SET &FNDFLG = YES
   SET &RCX = Ø
 /*WRITE FOUND JOB &substr(1:8,&JOBNAME) IN Excl LIST-CHK RC IS Nxt*/
   /*Check the RC field            */
   SET CHKRC1 = 99
   SYSCALL CHKFLD 2 &JRC
   IF &LASTCC = Ø THEN SET CHKRC1 = Ø
   IF &CHKRC1 = Ø THEN DO
     SET CHKRC2 = 99
   /*WRITE RETURN CODE FOR &SUBSTR(1:8,&JOBNAME) IS IN XCLUSION FILE*/
```

```
      SET JSTEP = &JSTEP&STR(        )
      SYSCALL CHKFLD 3 &JSTEP
      IF &LASTCC = Ø THEN SET CHKRC2 = Ø
      ENDO
    IF &CHKRC2 = Ø THEN DO
      SET CHKRC3 = 99
    /*WRITE JOBSTEP IS IN XCLUSION FILE - &JSTEP*/
      SET JPSTEP = &JPRSTEP&STR(        )
      IF &JPSTEP = &STR(        ) THEN SET JPSTEP = ?
      SYSCALL CHKFLD 4 &JPSTEP
      IF &LASTCC = Ø THEN SET CHKRC3 = Ø
      ENDO
    IF &CHKRC3 = Ø THEN DO
      SET CHKRC4 = 99
    /*WRITE PROCSTEP IS IN XCLUSION FILE - &JPSTEP*/
      SET JPROC = &JPROC&STR(        )
      IF &JPROC = &STR(        ) THEN SET JPROC = ?
      SYSCALL CHKFLD 5 &JPROC
      SET &RCCC = &LASTCC
    /*WRITE CALL to CHKFLD 5 GOT RC'&RCCC' */
      IF &RCCC = Ø THEN SET CHKRC4 = Ø
      ENDO
    IF &CHKRC4 = Ø THEN DO
    /*WRITE PROC IS IN EXCLUSION FILE - &JPROC */
      ENDO
    ENDO
IF &FNDFLG = YES THEN IF &CHKRC4 = 99 THEN GOTO CHECKJ2
CLOSFILE IN3
TSO FREE FILE(IN3)
IF &CHKRC4 = Ø THEN RETURN CODE(Ø)
IF &CHKRC4 = 99 THEN RETURN CODE(99)
ENDO
CHKFLD: PROC 2 FLDNUM FIELD
/* CHKFLD accepts two arguments - the relative field number in the*/
/* file and the variable to compare it with from IOF (jobname etc)*/
/* The zero return code means there was a match, 99 no match      */
/* SCOLS contains the starting columns of all checkrc file fields*/
/* SLENS contains the maximum length of    all checkrc file fields*/
/* WRITE ENTRY TO CHECKFLD FLDNUM'&FLDNUM' FIELD'&FIELD'*/
IF &FIELD = ? THEN SET &FIELD = &STR(        )
SET SCOLS= &STR(Ø11Ø15243342)
SET SLENS= &STR(Ø8Ø4Ø8Ø8Ø8Ø6)
SET OFFSET = (&FLDNUM * 2) - 1
SET OFFSET2 = &OFFSET + 1
/*WRITE OFFSET'&OFFSET' OFFSET2'&OFFSET2' */
SET FLDSTRT = &SUBSTR(&OFFSET:&OFFSET2,&SCOLS)
SET MAXLEN = &SUBSTR(&OFFSET:&OFFSET2,&SLENS)
SET COMPLEN = 1
SET SRCHLEN = 1
```

```
/*WRITE FLDSTRT'&FLDSTRT' MAXLEN'&MAXLEN'*/
/*Here we get the length of the field to use for the compare*/
DO WHILE &SRCHLEN <= &MAXLEN
   IF &SUBSTR(&FLDSTRT,&IN3)=&STR(*) THEN SET COMPLEN = &COMPLEN - 1
   IF &SUBSTR(&FLDSTRT,&IN3)=&STR( ) THEN SET COMPLEN = &COMPLEN - 1
/* WRITE COMPARE'&SUBSTR(&FLDSTRT,&IN3)' */
   SET &SRCHLEN = &SRCHLEN + 1
   SET &COMPLEN = &COMPLEN + 1
   SET FLDSTRT = &FLDSTRT + 1
   ENDO
SET FLDSTRT = &SUBSTR(&OFFSET:&OFFSET2,&SCOLS)
/*exclusion file had a * in this field so this is considered a match*/
IF &COMPLEN = 1 THEN RETURN CODE(Ø)
IF &COMPLEN > 1 THEN SET COMPLEN = &COMPLEN - 1
SET FLDEND = &COMPLEN + &FLDSTRT - 1
SET FIELD = &FIELD&STR(        )
/*WRITE FNUM'&FLDNUM' FIELD'&SUBSTR(1:&COMPLEN,&FIELD)' */
/*WRITE STRT'&FLDSTRT' FEND'&FLDEND' MAXLEN'&MAXLEN' */
/*WRITE CLEN'&COMPLEN' INFL'&SUBSTR(&FLDSTRT:&FLDEND,&IN3)' */
IF &COMPLEN ¬= &STR(*) THEN DO
   /*Here is where we compare the field in the file with the one */
   /*passed to this routine (gotten from IOF)                    */
   /*FIELD is from IOF, IN3 is from the exclusion file           */
   SET &FIELD = &FIELD&STR(       )
   IF &SUBSTR(&FLDSTRT:&FLDEND,&IN3) = &SUBSTR(1:&COMPLEN,&FIELD) -
   THEN DO
   /*WRITE THE COMPARISON WAS EQUAL */
     RETURN CODE(Ø)
     ENDO
   SET DATAT = &SUBSTR(&FLDSTRT:&FLDEND,&IN3)
   SET DATAT2 = &SUBSTR(1:&COMPLEN,&FIELD)
   IF &FLDNUM = 2 THEN DO
   /*WRITE PROCESSING FIELD NUMBER 2 */
     IF &DATATYPE(&DATAT) = NUM THEN DO
   /*WRITE EXCLUSION FILE INPUT IS NUMERIC*/
     IF &DATATYPE(&DATAT2) = NUM THEN DO
      /*WRITE FIELD FROM IOF IS NUMERIC */
     IF &DATAT> &DATAT2 THEN DO
     /* WRITE EXCLUSION FILE RC IS GREATER THAN IOF RC OF JOB */
        RETURN CODE(Ø)
        ENDO
        ENDO
        ENDO
     IF &DATATYPE(&SUBSTR(&FLDSTRT:&FLDEND,&IN3))= ABND THEN DO
      /*WRITE ABENDS ARE ACCEPTABLE FOR THIS JOB */
        RETURN CODE(Ø)
        ENDO
     ENDO
   IF &SUBSTR(&FLDSTRT:&FLDEND,&IN3) ¬= &SUBSTR(1:&COMPLEN,&FIELD) -
```

```
      THEN DO
    /*WRITE THE COMPARISON WAS NOT EQUAL */
      RETURN CODE(99)
      ENDO
    ENDO/*IF*/
/*WE SHOULD NEVER FALL THROUGH TO THIS CODE- ASSUME NO MATCH*/
RETURN CODE(99)
ENDO /*CHECKFLD PROC*/
```

SAMPLE INPUT FILE FORMAT FOR INCLUDE LIST (RCCHECK.INCLUDE)

```
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* THIS IS THE INCLUDE FILE USED BY THE CHECKRC ROUTINE TO DETERMINE
* WHICH JOBS TO CHECK IN THE JES2 OUTPUT QUEUE FOR BAD RETURN CODES.
* THE FIRST COLUMN IS THE JOBNAME WHICH CAN BE CODED WITH AN '*'
* SUFFIX TO INDICATE A JOBNAME THAT BEGINS WITH THE PREVIOUS LETTERS.
* THE SECOND COLUMN IS FOR THE PAGEID FOR PAGING. THE THIRD COLUMN IS
* THE MAILID FOR SENDING MAIL MESSAGES. THE FORTH COLUMN IS FOR THE
* WORD 'CALENDAR' WHICH INDICATES A CALENDAR FILE WILL BE USED. THE
* FIFTH COLUMN IS A CALENDAR FILE NAME USED TO DETERMINE MAILIDS AND
* PAGERIDS BASED ON THE DATE. THE CALENDAR FILE IS ONLY NECESSARY WHEN
* THERE ARE PEOPLE WHO ROTATE THEIR JOB MONITORING RESPONSIBILITIES
* BASED ON THE DATE. LOOK INSIDE A CALENDAR FILE TO DETERMINE ITS
* FORMAT.
*
* NOTE: THE TABLE BELOW IS COLUMN DEPENDENT. COMMENTS MUST START
*       WITH AN '*' IN COLUMN ONE.     !!!MORE DATA TO THE RIGHT ===>
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*COL1-8  COL1Ø-18  COL2Ø-58                           COL6Ø-67  COL7Ø-115
*JOBNAME PAGERID   MAILID
CALENDAR  NAME OF THE CALENDAR FILE TO USE FOR GROUP OF JOBS
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
SMFDUMP  IBMSYS    PRODSUPPORT@ABCD.ORG      CALENDER   RCCHECK.CALENDER
OPTMSBK  IBMSYS    SYSSUPPORT@ABCD.ORG
OPWDUMP  IBMSYS    SYSSUPPORT@ABCD.ORG
OPMDUMP  IBMSYS    SYSSUPPORT@ABCD.ORG
CICSIFC  IBMSYS    SYSSUPPORT@ABCD.ORG
HSM      IBMSYS    PRODSUPPORT@ABCD.ORG
OPAEXTD  IBMSYS    PRODSUPPORT@ABCD.ORG
OPBEXTD  IBMSYS    PRODSUPPORT@ABCD.ORG
EOF.....END OF FILE DELIMETER--DO NOT DELETE OR ALTER THIS LINE
```

SAMPLE INPUT FILE FORMAT FOR EXLUDE LIST (RCCHECK.EXCLUDE)

```
JOBNAME   RC     STEP    PROCSTEP    PROC    C/NOCALL COMMENTS
```

```
-------- ---- -------- -------- -------- -------- --------------------
EISPX*   4    LISTC    *        *        NOCALL
OPAEXTDP 4    *        *        *        NOCALL
OPAEXTDF 4    *        *        *        NOCALL
OPBEXTDP 4    *        *        *        NOCALL
OPBEXTDF 4    *        *        *        NOCALL
EOF.....END OF FILE DELIMETER--DO NOT DELETE OR ALTER THIS LINE
```

The following JCL can be used to run the above CLIST program (RCCHECK) in batch:

```
//RCCHECK  JOB (,OPR),'CHECK RET CODE',MSGCLASS=T,MSGLEVEL=(1,1),
//             REGION=1Ø48K
//*LOGONID SYSJOB
//*--------------------------------------------------------------
//*   CHECK RETURN CODES OF JOBS THAT FINISHED IN THE LAST HOUR
//*--------------------------------------------------------------
//*   THIS ROUTINE USES THE FOLLOWING FILES:
//*   RCCHECK.INCLUDE      - LIST OF JOBS TO CHECK
//*   RCCHECK.EXCLUDE      - LIST OF JOBS TO EXCLUDE
//*   RCCHECK.CALNDRØ1     - OPTIONAL CALENDAR FOR ALERTS
//*   IOF Version 7D or higher is needed
//*   Put RCCHECK clist program in SYS1.PROD.CLIST dataset
//CHECKIT EXEC PGM=IKJEFTØ1
//SYSPROC   DD  DISP=SHR,DSN=SYS2.IOF.IOFT7DØ.CLIST
//          DD  DISP=SHR,DSN=SYS1.PROD.CLIST
//STEPLIB      DD DSN=CHECK.ACTIVE.LOADLIB,DISP=SHR
//SYSTSPRT  DD  SYSOUT=*
//SYSPRINT  DD  SYSOUT=*
//OUTPUT    DD  SYSOUT=*
//SYSTCPD   DD  DSN=TCPIP.TCPPARMS(TCPDATA),DISP=SHR
//* THE FOLLOWING SYSIN IS REQUIRED FOR REXEC - DO NOT REMOVE
//SYSIN     DD  *
/*
//SYSTSIN   DD  *
%RCCHECK
/*
//
```

SAMPLE INPUT FILE FORMAT FOR EXLUDE LIST (RCCHECK.CALENDAR)

```
* DESCRIPTION:                                                     *
* THIS FILE IS USED BY THE CHECKRC CLIST. IT IS POINTED TO BY THE  *
* 'INCLUDE' FILE THAT IS USED TO LIST THE JOBS THAT ARE TO BE CHECKED *
* FOR BAD RETURN CODES. THE INCLUDE FILE CAN SPECIFY THE CALENDAR FILE*
* THAT IS USED FOR ANY JOB OR GROUP OF JOBS.                       *
*                                                                  *
* THIS CALENDAR FILE HAS ENTRIES BY DATE INDICATING WHO SHOULD BE      *
* PAGED OR EMAILED WHEN A NOTIFICATION ALERT IS TO BE SENT. IT IS      *
* BASED ON DATE.                                                       *
*                                                                      *
* NOTE: THE TABLE BELOW IS COLUMN DEPENDENT. COMMENTS MUST START       *
* NOTE: WITH AN '*' IN COLUMN ONE.                                     *
*                                                                      *
* NOTE: THE TABLE BELOW MUST BE IN ASCENDING DATE ORDER. THE DATE      *
* NOTE: IS IN COLUMN ONE IN THE FORMAT: YYMMDD                         *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*COL1-6  COL1Ø-17  COL2Ø-58                                COL6ØCOMMENT
*DATE    PAGERID   MAILID
Ø31116   IBMMVS2   PRODSUPPORT@ABCD.ORG
Ø31116   IBMMVS2   PRODSUPPORT@ABCD.ORG
```

ASSEMBLER PROGRAM (WTOCLIST)

```
         EJECT
TITLE 'WRITE TO CONSOLE FROM CLIST PROGRAM'
WTOCLIST CSECT
         USING WTOCLIST,R12
         STM   R14,R12,12(R13)
         LR    R12,R15
         ST    R13,SAVEAREA+4
         LA    R1Ø,SAVEAREA
         ST    R1Ø,8(R13)
         LR    R13,R1Ø
         B     MAINLINE
         DC    CL1Ø'WTOREXX'
         DC    CL1Ø'&SYSDATE'
         DC    CL1Ø'&SYSTIME'
MAINLINE DS    ØH
         LR    R2,R1
         L     R3,16(R2)
         L     R8,2Ø(R2)
         L     R8,Ø(R8)
         USING EVALBLOCK,R8
         LR    R2,R3                      ARGLIST
MOVMSG   LA    R6,MSG
         L     R3,4(R2)                   LENGTH
         L     R2,Ø(R2)                   MESSAGE
         LR    R7,R3
         STH   R3,MSGL
         MVCL  R6,R2
         LA    R7,MSGL
         WTO   TEXT=(R7),DESC=1      DECRIPTOR HIGHLIGHTS ON CONSOLE
         MVC   EVALBLOCK_EVLEN,=F'1'
         MVC   EVALBLOCK_EVDATA,=C'Ø'
```

```
*-----------------------------------------------------------------*
RETURN   DS    ØH
         L     R13,SAVEAREA+4    RESTORE R13
         LM    R14,R12,12(R13)   RESTORE R14 TO R12
         XR    R15,R15           ZERO RETURN CODE REG
         BR    R14               RETURN
*-----------------------------------------------------------------*
* EQUATES                                                         *
*-----------------------------------------------------------------*
RØ       EQU   Ø
R1       EQU   1
R2       EQU   2
R3       EQU   3
R4       EQU   4
R5       EQU   5
R6       EQU   6
R7       EQU   7
R8       EQU   8
R9       EQU   9
R1Ø      EQU   1Ø
R11      EQU   11
R12      EQU   12
R13      EQU   13
R14      EQU   14
R15      EQU   15
*-----------------------------------------------------------------*
SAVEAREA DC    18F'Ø' ADDRESSED BY REG 13
         EJECT
*-----------------------------------------------------------------*
MSGL     DS    H
MSG      DS    CL2ØØ
*-----------------------------------------------------------------*
*        EVALBLOCK DSECT                                          *
*-----------------------------------------------------------------*
         IRXEVALB
         END   WTOCLIST
```

K Muthukumar
Systems Programmer (USA)

Disaster recovery procedure – part 2

This month we conclude the code for a disaster recovery procedure.

```
/*******************************************************************/
/*  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,1Ø);
   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 > Ø
 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 nonvsam 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) = '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 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 * 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 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)

REXX routine to count lines of COBOL code

Recently I had various requests: to count the lines of COBOL code in all the members of a partitioned dataset; to edit all members of a PDS; to use ISPF in batch; etc.

I decided to combine the various requests, and have written a routine that addresses these regularly occurring problems. This useful routine counts the lines of COBOL code while demonstrating a few useful techniques in REXX/ISPF.

LIMITATIONS

The code counting routine counts the total number of lines in a program and then reduces this by the number of blank and comment lines (* in position 7). It does not count the number of COBOL statements and therefore when a statement goes over more than one line this will result in a positive discrepancy (ie lines of code > number of statements).

The routine will become time and CPU intensive when the size of the datasets to be scanned is large. It is therefore recommended that the routine is first tested on smaller sample datasets, check that the set up is correct, and then let the final scan run in an overnight batch window.

OPTIONS

The routine has various optional elements included to give a depth of flexibility to accommodate the requirements of the end user (report reader) and the report generator (JCL submitter). These elements are as follows:

1 Procedure division only – here the count is reduced to the lines of procedural code, ie the whole data division is ignored.

2 Expand copybooks – copybook entries found in the code may be expanded to show the 'real size' of a program or may be ignored to show the lines of non-reusable code. When this option is used the copybooks are expanded in up to three levels of nesting.

3 Should the expand copybook option be used, a copybook store (just the statistical information) may be used in one of three modes:

   - O (Output) – here the store is created.
   - I (Input) – here the store is used only on its own (copybooks are not further expanded by the routine).
   - U (Update) – the original values are used from the copybook store, but are further expanded should there be new values or further nested copy statements.

4 One source PDS may be specified, but up to six copybook libraries can be processed.

5 The report output dataset may be steered to produce a unique dataset per run, the name being dependent on date and time.

SUPPLIED CODE

The following code is provided in this article:

- JCL – BATCHPDF.
- ISPF edit macro – COBCMEM.
- REXX routine – COBCOUNT.

BATCHPDF is a REXX routine available from Doug Nadel (www.sillysot.com/mvs/).

INTERESTING TECHNIQUES

In developing this routine and addressing the various regularly occurring problems, the following techniques are demonstrated: ISPF (edit macros) in batch; accessing all members of a PDS; searching for a target greater than the command line; the REXX **interpret** command; testing for the existence of a dataset; allocating a dataset; restricting the code searched using labels; restricting the columns searched using the BOUNDS command; and a simple recursive call in REXX.

ISPF in batch

At first, as I was developing the routine, I was testing it in the foreground. When I started to use it background (batch) I was suddenly confronted with return code 990. My first reaction was to ask myself what was different, what had changed. After analysing the situation and scratching my head a lot, it was clear that my batch job was missing the ISPF environment and therefore the edit macro was failing. I asked my colleagues for a sample of JCL for batch ISPF and, as I was waiting for their response, had a look at the Internet. There I found a solution on

Doug Nadel's site – his REXX routine BATCHPDF. This was 95% right for what I wanted. The other 5% were my own modifications to the JCL output by the routine. His solution gave an inline CLIST and I changed this to an inline REXX procedure. The only other change was to include the BDISPMAX option on the ISPSTART line to allow displays for debugging:

```
ISPSTART CMD(%TEMPNAME) NEWAPPL(ISR) BDISPMAX(5ØØØØ)
```

When you have Doug Nadel's routine you need to make sure that it is in a library accessible from your TSO, and then in an empty member. Simply type BATCHPDF on the command line; this will produce a sample JCL with all your current ISPF allocations included.

Accessing all members of a PDS

Again, the Internet provided the quick answer here.

On Lionel B Dyck's site (www.lbdsoftware.com/ispftools.html) I found his edit macro, DOALL, to be of use and have incorporated the code into my routine. This code lists the member information from a dataset, skips the header information, and then processes one member after another using an edit macro.

Searching for a target greater than the command line

This was originally a simple edit macro I wrote to enable search strings greater than the length of the command line to be used for someone who wanted to search for blank lines of 80 characters. The logic I have simply included in my edit macro, COBCMEM:

```
/* REXX */                      /* required REXX identifier          */
/* FINDBL8Ø                                                          */
/* ***************************************************************** */
"ISREDIT MACRO NOPROCESS"       /* required EDIT MACRO identifier    */
"ISPEXEC CONTROL ERRORS RETURN"
trace o                         /* trace switch                      */
/* ***************************************************************** */
/* ***************************************************************** */
bl8Ø = ' '
bl8Ø = substr(bl8Ø,1,8Ø,' ')     /* make bl8Ø contain 8Ø blanks       */
    "ISREDIT FIND '"bl8Ø"' ALL" /* search for all blank 8Ø's         */
    "ISREDIT (allstr,allline) = FIND_COUNTS"
    say 'Total Occurrences of String = ' allstr
    say 'Total Number of lines where string is found = ' allline
return
```

The interpret command

As I was expanding the copybooks for later use in the COBCOUNT routine I was aware that I had unique names of copy members and values associated with these members. I thought about creating an array with the names and the values, comparing these with the names found in the COPY statements, and then adding these values to give a final count. This method would work, but would require a scan of the array for each COPY statement member found. After a huge amount of head scratching, I decided to try to use REXX's interpret command. This command is, I believe, unique to REXX and allows a command to be dynamically built and interpreted in one step. In my case I wanted to set values to variables whose names I didn't know until they were extracted from the names of the dataset members. Each time I had a final count value for a member, I built this together with the name of the member, and ran this through the interpret command to give me a variable with the name of the member set to a value extracted from the member itself. When it came to expanding further copybooks (or nested copy statements) or the source dataset, I needed only to quote the copy statement member and could simply add these together.

Testing for the existence of a dataset, allocating a dataset, and simple recursive call of a REXX subroutine

All three are found together in one subroutine and that is def_output in the COBCOUNT routine. Here I use the TSO external function SYSDSN to find the status of the dataset and ALLOC to allocate the new dataset. When the dataset already exists I delete it, call (recursively) the routine within itself, and then pass through the subroutine a second time to allocate the, then, new dataset.

Using labels and the BOUNDS command in an edit macro

These two features are found in the edit macro COBCMEM.

The BOUNDS command is used to bind/restrict the editor to a specific column range.

The first action is to record the boundary positions before changing them with:

```
 'ISREDIT (bnd1,bnd2) = BOUNDS'
```

bnd1 is the start boundary position and *bnd2* the end boundary position.

The second action is, in this case, to set the range to column 7:

```
 "ISREDIT BOUNDS = 7 7 "
```

Then search for '*', ie comment lines. After the search on column 7 the boundary values are changed back to their original values:

```
 "ISREDIT BOUNDS = " bnd1 bnd2 ""
```

The FINDs performed are qualified with labels for the start and end row positions to search. The labels used are .START and .END respectively. When the procedure division is being searched the .START is set to the line where PROCEDURE DIVISION is found and .END to the last line. When the Data Division is being searched .START is set to the first line of code and .END is set to the line prior to the line where PROCEDURE DIVISION is found.

Establish the position of PROCEDURE DIVISION; variable 'procsplt' will contain the line number:

```
'ISREDIT SEEK "PROCEDURE DIVISION" FIRST'
'ISREDIT (procsplt) = LINENUM .ZCSR'
```

1 Start is set to the line of procedure division and end to the last line:

```
'ISREDIT LABEL 'procsplt' = .START Ø'
'ISREDIT LABEL .ZLAST = .END Ø'
```

2 End is set to the line of procedure division and start to the first line:

```
'ISREDIT LABEL 'procsplt' = .END Ø'
'ISREDIT LABEL .ZFIRST = .START Ø'
```

Find command using labels:

```
"ISREDIT FIND '*' .START .END ALL"
```

EXTRA NOTES ON USE

When the copybook information store dataset is used, the PDO (Procedure Division Only) option should remain consistent – either always ‘Y’ or always ‘N’. This is because the already-expanded copybook members will have been expanded using one or the other and disprepancies will most probably result when a mix is used.

Should there be more than three levels of nested copybooks, these can be extracted by repeatedly running the routine with the ‘U’ option for the copybook information store until the result no longer changes, ie all levels of nesting have been expanded.

The copybook store contains a list of the expanded copybook values. When it is used with the U option it is simply added to. That means the same member name will have more than one value attributed to it, the last one in the list being the current value. Because I’ve been testing this on small datasets and the results are as expected (the last value per variable being used), I’ve not developed any automatic housekeeping routine. When I get that far I’ll probably use an intermediate sort (descending) and then discard the duplicates. Other alternatives are by hand or even a simple REXX routine.

BATCHPDF – JCL

```
//*------------------------------------------------------------------*
//*                       Create Startup CLIST                       *
//*------------------------------------------------------------------*
//GENERØ   EXEC PGM=IEBGENER
//SYSUT1   DD *
 /* REXX */
 trace o
 "ISPEXEC VGET (ZTIMEL)" /* ISPF services can run here */
 say ZTIMEL
 /* Add any setup here                                            */
 /* ********************************************************      */
 /* 1) COBOL source dataset goes here                             */
 /* 2) Expand COPYBOOKS  'Y' or 'N'                                */
 /* 3) PROCEDURE DIVISION ONLY   'Y' or 'N'                        */
 /* 4) Output Dataset 'U' Unique 'R' Rewrite                       */
 /* 5)  COPYBOOK INFO. STORE DATASET  "O" Output only              */
 /*                                   "I" Input only               */
 /*                                   "U" Update existing store    */
 /*                                   "N" don't use                */
 /* 5a) COPYBOOK INFO. STORE DATASET  Only enter if 'O','I' or 'U' */
 /* 6)-6a) COPYBOOKS up to 6 copybook datasets here */
 queue   "'AL13745.PRIVATE.SRC'"
 queue   "Y"
 queue   "Y"
 queue   "U"
 queue   "O"
 queue   "'AL13745.COPYBOOK.DATA'"
 queue   "'AL13745.PRIVATE.SRC'"
 /* ****************************************************** */
 "ISPEXEC SELECT CMD(COBCOUNT)"
 /*                                                       */
 /* To invoke the editor, code a line like:               */
 /*   ISPEXEC EDIT DATASET('FRED.CNTL(BUBBA)') MACRO(BINKY)  */
 /* and remember that the macro must do an END or CANCEL.   */
 /* You can set the step return code if you want.         */
 ZISPFRC = rc
 "ISPEXEC VPUT (ZISPFRC) SHARED"     /* set step return code */
//SYSUT2   DD DISP=(NEW,PASS),DSN=&&CLISTØ(TEMPNAME),
//            SPACE=(TRK,(1,1,2),RLSE),UNIT=SYSALLDA,
//            DCB=(LRECL=8Ø,BLKSIZE=Ø,DSORG=PO,RECFM=FB)
//PROFILE  DD DISP=(NEW,CATLG),DSN=AL13745.T513Ø11.ISPPROF,
//            SPACE=(TRK,(1Ø,1Ø,5)),UNIT=SYSALLDA,
//            DCB=(LRECL=8Ø,BLKSIZE=Ø,DSORG=PO,RECFM=FB)
//SYSPRINT DD DUMMY
//SYSIN    DD DUMMY
//*------------------------------------------------------------------*
//*             Initialize profile dataset (optional)                *
//*------------------------------------------------------------------*
//* COPY     EXEC PGM=IEBCOPY
//* SYSPRINT DD  DUMMY
//* SYSIN    DD  DUMMY
//* SYSUT1   DD  DISP=SHR,DSN=AL13745.ISPFALFØ.PROFILE
//* SYSUT2   DD  DISP=(OLD,PASS),DSN=*.GENERØ.PROFILE
//*------------------------------------------------------------------*
//*                          Invoke ISPF                             *
//*------------------------------------------------------------------*
//BATCHPDF EXEC PGM=IKJEFTØ1,DYNAMNBR=128
//ISPLLIB  DD DISP=SHR,DSN=AL13745.ISPF.LOAD                  ALUØØ2
//ISPPLIB  DD DISP=SHR,DSN=AL13745.ISPF.PANELS                ALUØØ2
//         DD DISP=SHR,DSN=SYS4.SPFSYS1.PANELS                SYSØØ3
//         DD DISP=SHR,DSN=SYS4.XUT11Ø.PANELS                 SYSØØ5
//         DD DISP=SHR,DSN=SYS4.SPF14ØA.SISPPENU.PANELS       SYSØ16
//         DD DISP=SHR,DSN=SYS4.XPF500.SXPFPANL.PANELS         SYS009
//ISPSLIB  DD DISP=SHR,DSN=AL13745.ISPF.SKELS                  ALU002
//         DD DISP=SHR,DSN=SYS4.SPFSYS1.SKELS                  SYS003
//         DD DISP=SHR,DSN=SYS4.XUT110.SKELS                   SYS005
//         DD DISP=SHR,DSN=SYS4.SPF140A.SISPSENU.SKELS         SYS016
//         DD DISP=SHR,DSN=SYS4.SPF140A.SISPSLIB.SKELS         SYS016
//ISPMLIB  DD DISP=SHR,DSN=AL13745.ISPF.MSGS                   ALU001
//         DD DISP=SHR,DSN=SYS4.SPFSYS1.MSGS                   SYS003
//         DD DISP=SHR,DSN=SYS4.XUT110.MSGS                    SYS005
//         DD DISP=SHR,DSN=SYS4.SPF140A.SISPMENU.MSGS          SYS016
//         DD DISP=SHR,DSN=SYS4.XPF500.SXPFMSG.MSGS            SYS005
//         DD DISP=SHR,DSN=SYS4.MVS140A.SBPXMENU.MSGS          SYS021
//ISPPROF  DD DISP=(OLD,PASS),DSN=AL13745.T513011.ISPPROF
//ISPTABL  DD DISP=(OLD,PASS),DSN=AL13745.T513011.ISPPROF
//ISPTLIB  DD DISP=(OLD,PASS),DSN=AL13745.T513011.ISPPROF
//         DD DISP=SHR,DSN=AL13745.ISPFALF0.PROFILE            ALU003
//         DD DISP=SHR,DSN=SYS4.XUT110.TABLES                  SYS005
//         DD DISP=SHR,DSN=SYS4.SPFSYS1.TABLES                 SYS003
//         DD DISP=SHR,DSN=SYS4.SPF140A.SISPTENU.TABLES        SYS016
//ISPCTL0  DD DISP=(NEW,DELETE),SPACE=(TRK,(10,10)),UNIT=VIO,
//            DCB=(LRECL=80,BLKSIZE=0,DSORG=PS,RECFM=FB)
//ISPCTL1  DD DISP=(NEW,DELETE),SPACE=(TRK,(10,10)),UNIT=VIO,
//            DCB=(LRECL=80,BLKSIZE=0,DSORG=PS,RECFM=FB)
//ISPWRK1  DD DISP=(NEW,DELETE),SPACE=(TRK,(10,10)),UNIT=VIO,
//            DCB=(LRECL=80,BLKSIZE=0,DSORG=PS,RECFM=FB)
//ISPLST1  DD DISP=(NEW,DELETE),SPACE=(TRK,(10,10)),
//            DCB=(LRECL=133,BLKSIZE=0,DSORG=PS,RECFM=VB)
//ISPLOG   DD SYSOUT=*,
//            DCB=(LRECL=120,BLKSIZE=2400,DSORG=PS,RECFM=FB)
//ISPLIST  DD SYSOUT=*,DCB=(LRECL=121,BLKSIZE=1210,RECFM=FBA)
//* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
 PROFILE PREFIX(AL13745)
 ISPSTART CMD(%TEMPNAME) NEWAPPL(ISR) BDISPMAX(50000)
//SYSEXEC  DD DISP=SHR,DSN=AL13745.ISPF.EXEC                   ALU001
//         DD DISP=SHR,DSN=SYS4.SPFSYS1.EXEC                   SYS003
//         DD DISP=SHR,DSN=SYS4.MVS140A.SBPXEXEC.EXEC          SYS021
//         DD DISP=SHR,DSN=SYS4.SPF140A.SISPEXEC.EXEC          SYS011
//SYSPROC  DD DSN=&&CLIST0,DISP=(OLD,DELETE)
//         DD DISP=SHR,DSN=AL13745.ISPF.CLIST                  ALU001
//         DD DISP=SHR,DSN=SYS4.SPFSYS1.CLIST                  SYS003
//         DD DISP=SHR,DSN=SYS4.XUT110.CLIST                   SYS005
//         DD DISP=SHR,DSN=SYS4.SPF140A.SISPCLIB.CLIST         SYS011
//*---------------------------------------------------------------*
//*                Delete temporary profile dataset                *
//*---------------------------------------------------------------*
//DELPROF  EXEC PGM=IEFBR14
//DELETEDD DD DSN=AL13745.T513011.ISPPROF,DISP=(OLD,DELETE)
```

COBCOUNT – REXX MAIN CODE

```
/* REXX */
/* ****************************************************************** */
/* ### COBCOUNT ### */
/* ****************************************************************** */
trace o
/* **************************************************************** */
/* * Initialization ************************************************* */
  editmac = 'COBCMEM'
/* **************************************************************** */
/* Definition variables for COPYBOOK info. store                      */
/*      "STORCLAS(BAØØ) MGMTCLAS(MALUØ###)",                          */
/*      "VOLUME(ALUØØ4) UNIT(339Ø)",                                  */
/*      "SPACE(2,5) TRACKS"                                           */
  stor1 = 'BAØØ'
  mgmt1 = 'MALUØ###'
  vol1  = 'ALUØØ4'
  unit1 = '339Ø'
  spc1  = '2,5'
/* **************************************************************** */
/* Definition variables for OUTPUT dataset                            */
/*      "STORCLAS(BAØØ) MGMTCLAS(MALUØ###)",                          */
/*      "VOLUME(ALUØØ4) UNIT(339Ø)",                                  */
/*      "SPACE(2,5) TRACKS"                                           */
  stor2 = 'BAØØ'
  mgmt2 = 'MALUØ###'
  vol2  = 'ALUØØ4'
  unit2 = '339Ø'
  spc2  = '2,5'
  outdsn = "'" || SYSVAR(SYSPREF) || ".OUT.TEMP" || "'"
/* **************************************************************** */
  srcdsn =''
  cpyexp =''
  pdo    =''
  outopt =''
  cbsuse =''
  cbsdsn = ''
  cpydsn.1=''
  cpydsn.2=''
  cpydsn.3=''
  cpydsn.4=''
  cpydsn.5=''
  cpydsn.6=''
  table1. = ''
  t1 = Ø
  cp = Ø
  drop memlist.
/* * Initialization *********************************************** */
/* **************************************************************** */
/* * get dataset names and control variables ********************** */
  pull srcdsn
  call check_srcdsn
  pull cpyexp
  pull pdo
  pull outopt
  /* if output dataset should be unique */
  if outopt = 'U' then
    do
      /* date and time stamp incorporated in dsn name */
      tlq = 'T' || time('S')
      dlq = 'D' || date('J')
      outdsn = SYSVAR(SYSPREF) || "."
      outdsn = outdsn || dlq || "."
      outdsn = outdsn || tlq || "."
      outdsn = outdsn || "OUT.TEMP"
      outdsn = "'" || outdsn || "'"
    end
  pull cbsuse
  /* U = Update (read in / write out) O = Output (Delete / Write out) */
  /* I = Input  (read in)                                             */
  if cbsuse = 'U' | cbsuse = 'O' | cbsuse = 'I' then
    do
      pull cbsdsn
      call check_cbsdsn
    end
  if cbsuse ¬= 'I' then
    do
      /* get the names of up to 6 copybook datasets */
      j1 = Ø
      do forever
        j1 = j1 + 1
        pull cpydsn.j1
        if cpydsn.j1 = '' then leave
        if j > 6 then leave
        cpydsn.j1 = strip(cpydsn.j1,"B")
        cpydsn = cpydsn.j1
        call check_cpydsn
      end
      cpydsn.Ø = j1 - 1
    end
  /* read the copybook info. store dataset */
  if cbsuse = 'I' | cbsuse = 'U' then
    call input_cbs
/* * get dataset names and control variables ********************** */
/* **************************************************************** */
  say 'srcdsn   =' srcdsn
  say 'cpydsn.1 =' cpydsn.1
  say 'cpydsn.2 =' cpydsn.2
  say 'cpydsn.3 =' cpydsn.3
```

```
 say 'cpydsn.4 =' cpydsn.4
 say 'cpydsn.5 =' cpydsn.5
 say 'cpydsn.6 =' cpydsn.6
 say 'cpyexp   =' cpyexp
 say 'pdo      =' pdo
 say 'cbsuse   =' cbsuse
 say 'cbsdsn   =' cbsdsn
 say 'outdsn   =' outdsn
```

Editor's note: this article will be concluded next month.

Rolf Parker
Systems Programmer (Germany)

MVS news

Axios has announced SmartAnalyzer for HSM, the latest version of its back-up reliability analysis software. SmartAnalyzer helps protect business data by identifying problems and inefficiencies in the way that data is managed by the HSM disk management system.

SmartAnalyzer for HSM can identify: missing back-ups, unusable back-ups and/or migrated datasets, back-ups taken by non-standard tools, conflicting back-up/ migration policies and statuses, excessive back-ups and migrations, and conflicting expiration date specifications.

For further information contact:
Axios Products, 1373-10 Veterans Highway, Hauppauge, NY 11788-3047, USA.
Tel: (631) 979 0100.
URL: http://www.axios.com/prod_smartanalyzer.html.

* * *

Candle Corporation has announced PathWAI Monitor for WebSphere MQ V110, a solution that provides new features and functionality to Candle's performance management solution for IBM WebSphere MQ.

PathWAI Monitor for WebSphere MQ enhancements include simplified configuration that accelerates WebSphere MQ implementation, Web browser-based support to manage WebSphere MQ from any computer, and expanded management capabilities for a 360-degree view of middleware environments.

PathWAI Monitor for WebSphere MQ V110 supports Microsoft Windows, HP-UX, HP NSK, AIX, Sun Solaris, OS/390, z/OS, AS/400 and Linux for Intel platforms.
For further information contact:
Candle, 100 North Sepulveda Boulevard, El Segundo, CA 90245, USA.
Tel: (310) 535 3600.
URL: http://www.candle.com/www1/cnd/portal/CNDportal_Channel_Master/0,2258,2683_2772,00.html.

* * *

Bus-Tech has announced zDASD, a new product for connecting mainframes to IBM's FAStT and NAS Gateway products in support of DASD (ECKD) file access.

zDASD connects to the mainframe using either FiCON or ESCON channels and can emulate up to sixty-four 3390 models 3, 9, or 27 storage volumes with alternate path support. The controller connects to IBM's FAStT arrays via Fibre Channel or to IBM's NAS Gateways via Gigabit Ethernet.

Data written by a mainframe to devices attached to zDASD is stored on the FAStT or NAS Gateway in such a way as to allow the data to be retrieved in exactly the same format as it was stored. zDASD makes it transparent to the mainframe hardware and operating system that IBM's open-system storage is being used rather than traditional mainframe DASD volumes. zDASD supports IBM, and compatible mainframes for the z/OS, MVS, VM and VSE operating environments.

For further information contact:
Bus-Tech, 129 Middlesex Turnpike, Burlington, MA 01803, USA.
Tel: (781) 272 8200.
URL: http://www.bustech.com/hot/fs/fs_20040115.htm.