In this issue

3  Code transfer from the Web to MVS
4  Profile of address spaces by UIC interval
7  Displaying IPL parameters
11 Synchronizing remote PDS members – part 2
18 A DASD migration guide
36 Year 2000 aid: replace source strings – part 2
62 Checking job datasets exist before job submission
72 MVS news
**MVS Update**

**Editor**  
Dr Jaime Kaminski

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

**Subscriptions and back-issues**  
A year’s subscription to *MVS Update*, comprising twelve monthly issues, costs £325.00 in the UK; $485.00 in the USA and Canada; £331.00 in Europe; £337.00 in Australasia and Japan; and £335.50 elsewhere. In all cases the price includes postage. Individual issues, starting with the January 1992 issue, are available separately to subscribers for £29.00 ($43.00) each including postage.

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

© Xephon plc 1998. All rights reserved. None of the text in this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, without the prior permission of the copyright owner. Subscribers are free to copy any code reproduced in this publication for use in their own installations, but may not sell such code or incorporate it in any commercial product. No part of this publication may be used for any form of advertising, sales promotion, or publicity without the written permission of the publisher. Copying permits are available from Xephon in the form of pressure-sensitive labels, for application to individual copies. A pack of 240 labels costs $36 (£24), giving a cost per copy of 15 cents (10 pence). To order, contact Xephon at any of the addresses above.

*Printed in England.*
Code transfer from the Web to MVS

When a colleague of mine recently downloaded an MVS Update article from the Xephon Web site to his PC and then uploaded it to his MVS system, he found to his disappointment that the program code would not run properly.

It was a REXX program, and, when he executed it, he received the following message:

IRXØØ13I Error running XXXXXXXX, line nn: Invalid character in program

This was rather puzzling, but a quick look at the code revealed that the offending character was a REXX ‘not’ (that is ^, in a ^= expression), which should be a hex value X'5F', but was instead a X'B0'. The REXX interpreter was rejecting this value. Another odd character turned out to be the ‘|’ operator, which should be X'4F', but was X'6A'.

Having discovered this, it was simple to code an ISPF edit macro to fix this and to cater for it in future uploads:

ISREDIT MACRO
ISREDIT CHANGE ALL X'BØ' X'5F'
ISREDIT CHANGE ALL X'6A' X'4F'
EXIT

The PC was running IBM Personal Communications 3270 Version 4.1 for Windows with an IEEE 802.2 connection to the host, code page 037. The upload was achieved using the IBM 3270 PC File Transfer Program for MVS/TSO Release 1.1.1 using the following command:

IND$FILE PUT XEPHFILE.TEXT ASCII CRLF RECFM(V) LRECL(133)

It seems that the ASCII to EBCDIC conversion taking place works fine for alphanumeric characters, but is suspect for unusual ones. Readers should be aware of this when transferring code.

Patrick Mullen
MVS Systems Consultant (Canada)  © Xephon 1998
Profile of address spaces by UIC interval

INTRODUCTION
My previous article ‘Unreferenced interval count distribution’, in MVS Update Issue 138, gives a breakdown of the global UIC values across all the central storage on the complex. The MVS Real Storage Manager also maintains a system of counts of UIC values for each address space on the system, and this gives a very quick and easy overview of executing jobs’ storage references.

In the Real Storage Manager Control and Enumeration Area (RCE), there are four fields, RCEFRV1 – 4, which are UIC range values set by the System Resources Manager. These correspond to fields in the Real Storage Manager Address Space Block Extension (RAX), RAXFBV1 – 4, which are the number of frames held by the address space in the intervals defined by the range values.

The REXX program ASUIC reads the range values in the RCE, then runs through the Address Space Vector Table (ASVT) to find all the active address spaces on the system, jumps to their Address Space Control Block, and then to their RAX. The information extracted is presented on an ISPF panel ASUICP in the form of an ISPF table which can be scrolled using standard PF keys.

As many address spaces are typically swapped out and thus use very few frames, I have included a threshold value which prevents the display of address spaces holding too few frames. The value of this variable, ‘fth’, is set to 1000, but this can be changed to whatever is appropriate in your installation. Finally, the ISPF table can be sorted according to any of the fields on the display, defaulting to address space name order.

ASUIC REXX
/*------------------------------ REXX--------------------------------*/
/* Function   : Profile of Address Spaces by UIC interval.           */
/*-------------------------------------------------------------------*/
numeric digits 21
fth = 1000; sort = 'A'; sortseq = 'jobn'
do forever
address ispexec,
   "tbcreate asutab names(jobn fbv1 fbv2 fbv3 fbv4 tfr),
   nowrite replace"
cvt  = storage(d2x(16),4)
rce  = storage(d2x(c2d(cvt)+c2d(x2c(Ø49Ø))),4)
frv1 = c2d(storage(d2x(c2d(rce)+c2d(x2c(Ø11c))),2))
frv2 = c2d(storage(d2x(c2d(rce)+c2d(x2c(Ø11e))),2))
frv3 = c2d(storage(d2x(c2d(rce)+c2d(x2c(Ø12Ø))),2))
frv4 = c2d(storage(d2x(c2d(rce)+c2d(x2c(Ø122))),2))
asvt = storage(d2x(c2d(cvt)+c2d(x2c(Ø22c))),4)
asvu = storage(d2x(c2d(asvt)+c2d(x2c(Ø2Ø4))),4)
maxu = c2d(asvu)
addr = d2x(c2d(asvu)+c2d(x2c(Ø210)))
asve = storage(addr,4)
do i = 1 to maxu
   unus = bitor(substr(asve,1,1),'7f'x)
   if unus = 'ff'x then
      nop
   else
      jbn = d2x(c2d(storage(d2x(c2d(asve)+c2d(x2c(Ø0ac))),4)))
      if jbn = Ø then
         jbn = d2x(c2d(storage(d2x(c2d(asve)+c2d(x2c(Ø0bØ))),4)))
      end
      jobn = storage(jbn,8)
      asn  = storage(d2x(c2d(asve)+c2d(x2c(Ø024))),2)
      rab  = storage(d2x(c2d(asve)+c2d(x2c(Ø178))),4)
      rax  = d2c(c2d(rab)+272)
      fbv1 = c2d(storage(d2x(c2d(rax)+c2d(x2c(Ø03c))),4))
      fbv2 = c2d(storage(d2x(c2d(rax)+c2d(x2c(Ø04Ø))),4))
      fbv3 = c2d(storage(d2x(c2d(rax)+c2d(x2c(Ø044))),4))
      fbv4 = c2d(storage(d2x(c2d(rax)+c2d(x2c(Ø048))),4))
      tfr  = fbv1 + fbv2 + fbv3 + fbv4
      if tfr > fth then; do
         address ispexec "tbadd asutab"
      end
   addr = d2x(x2d(addr)+4)
asve = storage(addr,4)
end
select
   when sort = 'A' then
      sortseq = 'jobn'
   when sort = '1' then
      sortseq = 'fbv1,N,D'
   when sort = '2' then
      sortseq = 'fbv2,N,D'
   when sort = '3' then
      sortseq = 'fbv3,N,D'
when sort = '4' then
  sortseq = 'fbv4,N,D'
when sort = 'T' then
  sortseq = 'tfr,N,D'
otherwise
  sortseq = 'jobn'
end
address ispexec "tbtop asutab"
address ispexec "tbsort asutab fields("sortseq")"
address ispexec "tbdispl asutab panel(asuicp)"
if rc ≠ Ø then
do
  address ispexec "tbclose asutab"
  exit
end
address ispexec "vget (sort fth)"
address ispexec "tbclose asutab"
end
exit

ASUICP PANEL

)attr
  ! type(output) color(green) just(left)
  # type(output) color(yellow) just(right)
  $ type(output) intens(high)
  " type(text) color(turq)
  type(text) skip(on) intens(low)
)body expand(@@)
%@-@ Address Spaces by UIC interval @-@
%COMMAND -->_ZCMD                                              %SCROLL -->_AMT
+ %Sort -->_Z"(A/1/2/3/4/T) %Frame threshold -->_Z +
  Space ] #frv1"] #frv2"] #frv3"] #frv4"] Fr
 " --------------------------[--------------------------]
)model
" !Z "]Z "]Z "]Z "]Z "]Z "]Z "
)init
  .help = asuich
  .zvars = '(sort fth jobn fbv1 fbv2 fbv3 fbv4 tfr)'
  &zcmd = &z
  &zttdmark = ' '
  if (&sort = ' ')
    &sort = 'A'
  if (&fth = ' ')
    &fth = '1000'
)proc
  vput (sort fth)
)end
Displaying IPL parameters

INTRODUCTION
The following REXX displays parameters set at IPL time. The IHAIPA control, which this REXX scans, can give you many other system settings, such as master catalog name.

/* REXX */
cvt = c2x(storage(10,4))
cvtext = c2x(storage(d2x(x2d(cvt)+140),4))
cvtipa = c2x(storage(d2x(x2d(cvtext)+392),4))
ipasys = d2x(x2d(cvtipa)+2152)
ipasym = storage(d2x(x2d(cvtipa)+288),8)
ipahwnam= storage(d2x(x2d(cvtipa)+24),8)
ipalpnam= storage(d2x(x2d(cvtipa)+32),8)
ipapldsn= storage(d2x(x2d(cvtipa)+416),44)
ipaplvol= storage(d2x(x2d(cvtipa)+461),6)
say 'Hardware name   ' ipahwnam
say 'LPAR name       ' ipalpnam
say 'Parmlib dsn & volser' strip(ipapldsn)', ipaplvol
say 'Symbols member suffix parameter' ipasym
ptr     = ipasys
call setvars
do loop = 1 to arg.Ø
    call sortout
    call retrieve
    call parser
    call say_it
end

exit_point:
exit Ø

sortout:
ipapdesa= c2x(storage(ptr,4))
ipapdesl= c2d(storage(d2x(x2d(ptr)+4),2))
ipapdedo= storage(d2x(x2d(ptr)+6),2)
checkedo = c2x(ipapdedo)
ptr = d2x(x2d(ptr)+8)
return

setvars:
/*
   vars taken from ihaipa dsect
*/
arg.1   = 'ALLOC  '
arg.2   = 'APF    '
arg.3   = 'APG    '
arg.4   = 'BLDL   '
arg.5   = 'BLDLF  '
arg.6   = 'CLOCK  '
arg.7   = 'CLPA   '
arg.8   = 'CMB    '
arg.9   = 'CMD    '
arg.10  = 'CON    '
arg.11  = 'CONT   '
arg.12  = 'COUPLE '
arg.13  = 'CPQE   '
arg.14  = 'CSA    '
arg.15  = 'CSCBLOC'
arg.16  = 'CVIO   '
arg.17  = 'DEVSUP '
arg.18  = 'DIAG   '
arg.19  = 'DUMP   '
arg.20  = 'DUPLEX '
arg.21  = 'EXIT   '
arg.22 = 'FIX'
arg.23 = 'GRS'
arg.24 = 'GRSCNF'
arg.25 = 'GRSRNL'
arg.26 = 'ICS'
arg.27 = 'IOS'
arg.28 = 'IPS'
arg.29 = 'LNK'
arg.30 = 'LNKAUTH'
arg.31 = 'LOGCLS'
arg.32 = 'LOGLMT'
arg.33 = 'LOGREC'
arg.34 = 'LPA'
arg.35 = 'MAXCAD'
arg.36 = 'MAXUSER'
arg.37 = 'MLPA'
arg.38 = 'MSTRJCL'
arg.39 = 'NONVIO'
arg.40 = 'NSYSLX'
arg.41 = 'NUCMAP'
arg.42 = 'RESERVD'
arg.43 = 'OPI'
arg.44 = 'OPT'
arg.45 = 'PAGE'
arg.46 = 'PAGE'
arg.47 = 'PAGNUM'
arg.48 = 'PAGTOTL'
arg.49 = 'PAK'
arg.50 = 'PLEXCFG'
arg.51 = 'PRODP'
arg.52 = 'PROG'
arg.53 = 'PURGE'
arg.54 = 'RDE'
arg.55 = 'REAL'
arg.56 = 'RER'
arg.57 = 'RSU'
arg.58 = 'RSVNONR'
arg.59 = 'RSVSTRT'
arg.60 = 'SCH'
arg.61 = 'SMF'
arg.62 = 'SMS'
arg.63 = 'SQA'
arg.64 = 'SSN'
arg.65 = 'SVC'
arg.66 = 'SWAP'
arg.67 = 'SYSNAME'
arg.68 = 'SYSP'
arg.69 = 'VAL'
arg.70 = 'VIODSN'
arg.71 = 'VRREGN'
arg.Ø = 71
return
parser:
select
    when (checkedo = 'ØØØØ') then,
        status = 'Default setting used'
    when (checkedo = 'FFFF') then,
        status = 'OP provided parm value'
    otherwise status = 'Parmlib('

ipapdedo

')'
end
return

say_it:
if parm.Ø = 1 then,
    say arg.loop parm.1 status
else do
    pad = copies(' ',length(arg.loop))
    say arg.loop parm.1 status
    do loop1 = 2 to parm.Ø
        say pad parm.loop1
    end
end
return

retrieve:
length = 30
parms = storage(ipapdesa,ipapdes1)
count = Ø
if ipapdes1 = Ø then do
    parm.1 = copies(' ',length)
    parm.1 = overlay(parms,parm.1,1,length,' ')
    parm.Ø = count + 1
    signal retrieve_exit
end
y1 = (ipapdes1/length)
yy = y1*length
do loop2 = 1 to yy by length
    count = count + 1
    parm.count = substr(parms,loop2,length)
end
parm.Ø = count
retrieve_exit:
return
This month we conclude our look at a utility that allows the contents of partitioned datasets to be maintained over multiple systems. The ISPF Panel SYCCP1 below has been repeated in its entirety for convenience; please note this when downloading code from our Web site.

**ISPF PANEL SYNCPI**

```ispf
%———%PDS SYNCHRONISATION%———+

%OPTION ———>_ZCMD
+
%1$ TRANSMIT+- Send changed members since last snapshot and take new snapshot
%2$ NEW SNAP+- Record a snapshot of member statistics (replace existing snap)
%3$ OLD SNAP+- Rename previous snapshot to recorded snapshot (if xmit failed)
%4$ DIS SNAP+- Display recorded snapshot member statistics (if any)
%5$ DEL SNAP+- Delete the recorded snapshot member statistics (if any)
+
+NOTE : Only members with ISPF (or PDSMAN) statistics will be processed.
+Specify 'DATASET' in TSO syntax (do not specify a member name)
+%DATASET NAME %===>_SDSN +(On both systems)
+Review xmit %===>_SLST+(Y/N, to review changed list before transmit)
+
+To+MVS node %===>_STOS + (second job will execute here)
+
+Batch JOBs+(jobname &ZUSER.XF/R) First job generates xmit, second receives it
+JOB Acct %===>_SACT +
+JOB Class %===>_SECL+ JOB msgclass %===>_SMCL+
+
+Enter details and press%ENTER+to continue or press%END+to exit

)INIT

HELP = SYNCPI

&ZCMD = '

.CURSOR = ZCMD

)PROC

VER (&ZCMD,NB,LIST,1,2,3,4,5)
VER (&SDSN,NB,DSNAME)
IF (&ZCMD = 1)
   VER (&SLST,NB,LIST,Y,N)
   VER (&STOS,NB)
   VER (&SACT,NB)
ISPF PANEL SYNCP2

%TUTORIAL ——————%PDS SYNCHRONISATION%—————— TUTORIAL+
%COMMAND —> _ZCMD
+
+ Caution: Multiple users on the same dataset simultaneously is not supported.
+
%TRANSMIT+- This is used to bring the datasets back into line, sending updates.
%NEW SNAP+- Use after a complete transfer (BDT/NFT) to start recording changes.
%OLD SNAP+- Use if a transmit fails, to regenerate the last update transfer.
%DIS SNAP+- See if a snapshot exists (and when taken) by browsing it.
%DEL SNAP+- Delete the snapshot if it is no longer required.
+
%To+system is the remote MVS JES node name of the system concerned.
+
%From dataset,+this must be a PDS (not RECFM U), same name on both systems.
%Review list,+allows the updated member names to be checked before transmit.
+
+ Account code and job class/msgclass values are self-explanatory.
+
+
+Press%END+to return
)PROC
 &ZCONT = SYNCP2
)END

ISPF PANEL SYNCP3

)ATTR
   $ TYPE(TEXT)   INTENS(LOW) COLOR(YELLOW)
   # TYPE(OUTPUT) INTENS(LOW) JUST(ASIS) COLOR(TURQ)
)BODY
%———————————%PDS SYNCHRONISATION%———————————+
%COMMAND —> _ZCMD %SCROLL —> _AMT +
%
%DATASET —> &SDSN +| PRESS%END/PF3+TO TRANSMIT |
+&MSG +| ENTER%'CANCEL'+TO ABORT |
%
%RESPOND QUICKLY+AS THE DATASET IS LOCKED FOR SYNC UPDATE DURING THIS DISPLAY %
%MEMBER DATE TIME USER
+
)MODEL
#2
)INIT
'ZVARS = '(INFO)'
&AMT = &ZSCML
IF (&SNAPDATE = ' ')
  &MSG = 'ALL MEMBERS LISTED, FULL TRANSFER'
ELSE
  &MSG = 'CHANGES SINCE &SNAPDATE &SNAPTIME'
)END

ISPF PANEL SYNCP4

)ATTR
  $ TYPE(TEXT) INTENS(LOW) COLOR(YELLOW)
  # TYPE(OUTPUT) INTENS(LOW) JUST(ASIS) COLOR(TURQ)
)BODY
%——————————-%PDS SYNCHRONISATION%——————————+%COMMAND ———> ZCMD
  %SCROLL ———> AMT +
%DATASET —> &SDSN +|
+&MSG
  PRESS%END/PF3+TO RETURN |
%MEMBER DATE TIME USER
  +———————————————————————————————————————-
)MODEL
#Z
)INIT
.ZVARS = '(INFO)'
&AMT = &ZSCML
IF (&SNAPDATE = ' ')
  &MSG = 'ALL MEMBERS LISTED, NO SNAPSHOT EXISTS'
ELSE
  &MSG = 'SNAPSHOT TAKEN &SNAPDATE &SNAPTIME'
)END

ISPF PANEL SYNCP5

)ATTR
  # TYPE(INPUT) INTENS(NON) CAPS(ON) JUST(LEFT)
  $ TYPE(TEXT) INTENS(LOW) COLOR(YELLOW)
)BODY
%——————————-%PDS SYNCHRONISATION%——————————+%COMMAND ———> ZCMD
  +
%DEL SNAP+- Delete the recorded snapshot member statistics (if any)
  +
+DATASET %&SDSN
  +
+Confirm that you wish to proceed with this option.
  +
+Press%ENTER+to continue or press%END+to return
ISPF SKELETON

)CM
)CM PDS SYNC JCL SKELETON
)CM
//&USER.XF JOB &SACT,'&USER',
   // NOTIFY=&USER,
   // CLASS=&SECL,MSGCLASS=&SMCL
   //*
   /** GENERATE FILE TRANSFER ON LOCAL FOR REMOTE SUBMIT
   /**
   //STEPX EXEC PGM=IKJEFTØ1,DYNAMNBR=9Ø,REGION=4M
   //SYSPROC DD DSN=SYS1.REXX.LIB,DISP=SHR
   //SYSTSIN DD SYSOUT=* 
   %SYNC2 STOS=&STOS -
   SDSN=&SDSN -
   USER=&USER -
   SECL=&SECL -
   SMCL=&SMCL -
   SACT=&SACT
   /*
   //MEMBERS DD *
)DOT &TABLE
&INFO
)ENDDOT
/*

ISPF MESSAGE MEMBER SYNC00

SYNCØØ1 '&&STATUS' .ALARM=YES
'DATASET &SDSN INVALID DUE TO &STATUS'

SYNCØØ2 'TRANSMIT JOB SENT' .ALARM=YES
'DATASET &SDSN TRANSMIT IN PROGRESS'

SYNCØØ3 'NOT A VALID PDS' .ALARM=YES
'DATASET &SDSN IS NOT A PDS OR HAS RECFM U'

SYNCØØ4 'LISTDSI FAILED' .ALARM=YES
ASSEMBLER PROGRAM SYNCL1

TITLE 'SYNCL1 ENQ/DEQ SUPPORT MODULE'

*****************************************************************
*  SYNCL1
*  THIS PROGRAM IS CALLED BY A FILE TRANSFER RECEIVE
*  REXX TO ENQ OR DEQ ON A DATASET.
*  CALLED MODULES (LINK TOGETHER WITH ENTRY POINT SYNCL1):
*  SYNCL2
*  NOT RENT, AC=Ø, R24
*****************************************************************

SYNCL CSECT
SPELEVEL SET=2  
STM R14,R12,12(R13) SAVE REGISTERS  
LR R12,R15 ADDRESSABILITY  
USING SYNCL1,R12 BASE REG
LR    R7,R1  PARM REG
GETMAIN R,lv=worklen  GETMAIN DYNAMIC AREA
LR    R10,R1  R10 -> DYNAMIC AREA
USING workarea,r10  ADDRESS DYNAMIC AREA
ST    R13,savearea+4  SAVE CALLERS SAVEAREA ADDRESS
ST    R10,8(r13)  SAVE SAVEAREA ADDRESS
LR    R13,R10  SAVE AREA PTR
MVC   eretc,=f'0'  RC
* PARM RETRIEVAL REXX LINK FORM
LM    R2,R3,0(r7)  LOAD ADDR/ADDR AND ADDR/LEN OF PARM
L     R2,0(r2)  LOAD ADDR
L     R3,0(r3)  LOAD LEN
SH    R3,=h'5'  MINUS 5 (4 + 1 FOR EX)
EX    R3,mover  MOVE DSNAME TO RNAME
* CHECK OPTION OF ENQ/DEQ
CLC   0(3,r2),=cl3'enq'  IS IT ENQ
BE    doenq
CLC   0(3,r2),=cl3'deq'  IS IT ENQ
BE    dodeq
TPUT  =cl4'r'sync1 parm NOT ENQ OR DEQ',40
MVC   eretc,=f'4'  RC
B     exit
Mover  MVC   RNAME(0),4(r2)  MOVE DSNAME
* ISSUE ENQ
Doenq  EQU   *
    CALL   syncl2,(enq,qname,rnamel,rname)  ISSUE ENQ
ST    R15,eretc
B     exit
* ISSUE DEQ
Dodeq  EQU   *
    CALL   syncl2,(deq,qname,rnamel,rname)  ISSUE DEQ
ST    R15,eretc
B     exit
* LEAVE PROGRAM
Exit   EQU   *
    L     R3,eretc  RETURN CODE
    L     R13,savearea+4  RESTORE R13
Freemain R,lv=worklen,a=(10) FREE WORK AREA
LR    R15,R3  RC
L     R14,12(r13)  RESTORE R14
LM    R0,R12,20(r13)  RESTORE R0 TO R12
BR    R14  RETURN
* STATIC STORAGE
Blanks  dc   cl255' '  BLANKS
Enq     dc   cl3'ENQ'
Deq     dc   cl3'DEQ'
Qname   dc   cl8'spfedit'
Rnamel  dc   al1(44)
Rname   dc   cl44' '
* UNINITIALIZED STORAGE (GETMAINED)

WORKAREA DSECT GETMAINED STORAGE AREA
SAVEAREA DS 18F SAVE AREA
ERETC DS F RC
WORKLEN EQU *-WORKAREA

YREGS
PRINT NOGEN
END SYNCL1

ASSEMBLER PROGRAM SYNCL2 (LINK WITH SYNCL1)

**********************************************************************
* SYNCL2 - SUBROUTINE TO ISSUE A ENQ OR A DEQ
* CALLING PARAMETERS:
*
* OPTION    CL3   EG ENQ, DEQ
* QNAME     CL8   EG SPFEDIT (FOR ISPF COMPATIBILITY)
* RNAME LEN XL1   EG 44
* RNAME     CL255  EG DSNAME OF 44
*
* LINK SYNCL1 AND SYNCL2 TOGETHER WITH ENTRY POINT SYNCL1
**********************************************************************

SYNCL2   CSECT
SYNCL2   AMODE 24
SYNCL2   RMODE 24
SAVE  (14,12),T     SAVE REGS
LR    R12,R15       LOAD BASE
USING SYNCL2,R12    USING
ST    R13,SAVEA+4   STORE FOWARD
LA    R11,SAVEA     ADDR OF MY SAVEAREA
ST    R11,8(R13)    STORE BACKWARD
LR    R13,R11       MY SAVE AREA PTR
LR    R9,R1         SAVE PARAM REG
L     R3,Ø(R9)      LOAD ADDR OF DDNAME
* ISSUE ENQ OR DEQ
L     R7,4(R9)      LOAD ADDR OF QNAME PARM
MVC   QNAME(8),Ø(R7)   COPY
L     R7,8(R9)      LOAD ADDR OF RNAME LEN
MVC   RNLEN+3(1),Ø(R7) COPY
L     R7,RNLEN      LOAD RNAME LEN
STC   R7,RNAME      STORE IN MACRO PARM
L     R6,12(R9)     LOAD ADDR OF RNAME
BCTR  R7,Ø          PREPARE FOR EX
EX    R7,MOVERNAM   MOVE RNAME
CLC  Ø(3,R3),=CL3'DEQ' DEQ
BE    ISSUEDEQ ISSUE DEQ
ENQ   (QNAME,RNAME,E,Ø,SYSTEMS)   ENQ IT
B    RETURN

ISSUEDEQ EQU *
A DASD migration guide

INTRODUCTION

As the ‘DASD farm’ becomes an increasingly important component of the IT department, a substantial proportion of systems programmers’ time can be dedicated to disk migration. Although 3390 disk geometry is now standard, migration from 3380 to 3390-style disks is not uncommon. We’ll examine the methods and the pitfalls of any DASD migration.

DFDSS

DFDSS is the instrument of choice to automatically move files from disk to disk. DFDSS COPY offers two kinds of processing:

- Physical – tracks are copied from the source volume to the target volume (of course, tracks not allocated are not copied). This is the fastest method, ideal when device types are the same (or differ only by capacity, the target being the largest). The source volume can be in use (in read mode) while it is copied. Our experience shows that up to 100 gigabytes can be moved in an hour (with state of the art disks).
• Logical – tracks (or records, or blocks) of allocated files are copied from the source volume to the target volume. Logical copy is mandatory with unlike device geometry (3380 to 3390), which is the most problematic situation. Only unallocated files can be moved to another volume. DFDSS in general does all the work; system utilities are invoked in a few cases for unlike device types (IEBCOPY for loadlibs, or IDCAMS for KSDS). IDCAMS (EXPORT/IMPORT) is always used for user catalogs. Rare utilities like IEHMOVE and IEBISAM are called when required.

A very useful precaution is to run in advance the DFDSS jobs with the parameter PARM=‘TYPRUN=NORUN’, to detect both syntax errors and potential problems with datasets.

DFDSS PHYSICAL MOVE
For copying the contents of volume xxx to volume yyy, this is the JCL I would recommend:

```jcl
//CPxxx EXEC PGM=ADRDSUU
//SYSPRINT DD SYSOUT=* 
//FROM1 DD VOL=SER=xxx,DISP=SHR,UNIT=SYSALLDA 
//TO1 DD VOL=SER=yyy,DISP=SHR,UNIT=SYSALLDA 
//SYSIN DD * 
COPY INDD(FROM1) OUTDD(TO1) FULL ALLEXCP - 
TOL(IOERROR) COPYVOLID
```

COPYVOLID ensures that the volume serial number from the source volume will be copied to the target volume. As two on-line volumes cannot bear the same name, the target volume will be set off-line as soon as the copy is finished. So to continue, you will have to:

1. Set the source volume OFF-LINE (this is only possible if no job is currently allocating it).
2. Set the target volume ON-LINE.
3. Reinitialize (ICKDSF INIT) or reformat (ICKDSF RFMT) the source volume.

Should step 3 not be performed (eg because step 1 cannot be performed), the following message will be received during the next IPL:

IEA213A DUPLICATE VOLUME vvv FOUND ON DEVICES dd1 AND dd2
One must then reply with the device number of the disk not to be used afterwards. Your operators will not thank you if they have not been previously informed. Unfortunately, you cannot avoid this scenario when handling some system volumes that can never be set off-line (SYSRES, mastercat volume, etc).

ALLDATA(*) and ALLEXCP are used to copy all the allocated space (even if not used). This is useful for files like the JES spool (that may appear empty to the system, but are not).

DFDSS LOGICAL MOVE

For moving all files from volume xxx to volume yyy, this is the JCL I would recommend:

```
//CPxxx      EXEC PGM=ADRDSSU
//SYSPRINT   DD   SYSOUT=* 
//FROM1      DD   VOL=SER=xxx,DISP=SHR,UNIT=SYSALLDA
//TO1        DD   VOL=SER=yyy,DISP=SHR,UNIT=SYSALLDA
//SYSIN      DD  *
  COPY LOGINDD(FROM1) OUTDD(TO1) TOL(IOERROR)  -
  ALLDATA(*) ALLEXCP ADMIN PURGE              -
  DELETE RECAT(*) ALLMULTI WAIT(0,0)          -
  DS(INCL(**),EXCL(SYS1.VTOCIX.*,SYS1.VVDS.*)) PROCESS(UNDEF)
```

Subparameters used:

- **ALLDATA(*)** – copy all the allocated space (files not empty).
- **ALLEXCP** – copy all the allocated space (empty files); DFDSS will not do it in all cases (IBM provides a table about ALLDATA and ALLEXCP interactions).
- **ADMIN** – to avoid any security problems with moving files (you must be authorized with only READ access to the STGADMIN.ADR.STGADMIN.COPY.DELETE profile in the FACILITY class).
- **PURGE** – you can override unexpired files on the target volume.
- **ALLMULTI** – catalogued multi-volume files are copied in their entirety (the risk being filling up the target disk).
- **WAIT(0,0)** – do not waste time waiting for files that are in use (and won’t be moved anyway).
• PROCESS(UNDEF) – copy (to unlike target disk) files with undefined DSORG.

If the source volume is an SMS-managed one, the ‘/TO1’ card should be omitted, as should the OUTDD parameter. The job should be preceded by a ‘V SMS,VOL(xxx),D,N’ command to disable new allocations for this volume. Sub-parameters to consider in some cases are:

• PROCESS(SYS1) – copy files with a high-level qualifier of SYS1.

• SPHERE – copy all AIX related to the base clusters you want to move (not recommended AIX may be copied even if they are not on the volume you want to empty – this may generate a no space left condition on the target disk).

• BYPASSACS(**) NULLSTORCLAS – to bypass SMS rules (this is not generally recommended).

The system parameters that need to be updated when disks or datasets are moved include:

• IODF and IOCP – of course, addresses for new units must be defined. Always keep old addresses for some time, in case of fallback.

• LOADxx in PARMLIB or IPLPARM – when the master catalog has been moved.

• SYSCATLG (or SYSCATxx) member of SYS1.NUCLEUS – if you still use it and the master catalog has been moved.

• EDT – if esoteric names are used in your JCL, there must be sufficient volumes must be linked to them so as to avoid JCL errors (EDT changing is now dynamic with HCD). It is simpler to adapt the EDT than to update the 'UNIT'= parameter in thousands of JCL files!

• APF list – to be updated, whether it be IEAAPFxx (static) or PROGxx (dynamic) in PARMLIB, every time an APF-authorized dataset was moved. No need to modify it for SMS-managed libraries or libraries located on the system resident volume (generic entries).
- VAT list (VATLSTxx) – for volume use attributes (use generic entries whenever possible). Often a co-requisite with EDT update, because disks used for non-specific allocation must be declared with the storage or public attribute.

- SMS parameters – new SMS volumes must belong to declared storage groups.

- HSM parameters (ARCCMDxx) – new primary (or migration, or backup) disk volumes must be declared (ADDVOL command), migrated ML/1 disks may be suppressed (DELVOL command), volume pools must be updated.

- Any program using dynamic allocation can be impacted, including some products tables (DSNZPARM with DB2), and some definitions (STOGROUPs for DB2, etc).

- IPL parameters – system-resident disk address and IPLPARM/IODF disk address (the loadparm parameter) can be affected by DASD migration.

- IPL texts – they are copied by the DFDSS COPY FULL function. Consider reinstalling them only after logical copies (SYSRES volume, stand-alone dump volume).

ON-LINE SYSTEM FILE MIGRATION

There are some general considerations, when a disk full copy cannot be undertaken. Many datasets require an IPL for MVS to acknowledge that they were moved (SYS1.SVCLIB, SYS1.STGINDEX, page datasets like PLPA or COMMON, SYS1.UADS, mastercat), while others are used only during the IPL (SYS1.LPALIB, SYS1.NUCLEUS, IODFs, IPLPARM).

In the recent versions of MVS, very few files still reside on the system resident volume (only three: SYS1.SVCLIB, SYS1.NUCLEUS and the optional PASSWORD file). When preparing to move system files, a precaution is to run the old IPO utility MCNVTCAT to be able to rebuild entries in the master catalog.
Direct access datasets

It is useful to know how direct access datasets are accessed. This can either be by track-track record (TTR), which is what DFDSS assumes by default, or by relative block address (less frequent). In the first case, DFDSS processes the file track by track (COPY TTRADDRESS parameter), and in the second, this is achieved block by block (COPY RELBLOCKADDRESS parameter). In the TTR case, DFDSS verifies that a target track can contain all the blocks of a source track (this is not obvious when disks differ by geometry). CICS, IDMS, and other DBMSs may use BDAM files.

Unmovable datasets

Usually, datasets are declared unmovable when DEFRAGing or HSM-migrating is to be avoided. Datasets with CCHHR location-dependent data should no longer exist. Nevertheless, DFDSS will try to place unmovable datasets at the same track locations on the target volume (which must be a like device), except if the FORCE parameter was specified (then DFDSS treats them as movable).

Model DSCB datasets

If you have used model DSCB datasets (instead of having a catalogued file used as a general model for all GDGs, or having SMS manage the GDGs), do not forget they must reside on the same volume as the related catalog.

JES2 spool

Initialize a spare disk (its name must match the SPOOLDEF VOLUME parameter). Allocate a spool file (name it according to the SPOOLDEF DSNAME parameter), whose size will be at least equal to the size of the active one(s). There is no need to catalogue it. Verify the value of the SPOOLNUM and TGSPACE (or TGNUM prior to SP510) parameters. Issue a ‘$SSPL,V=newvol,FORMAT’ command. When the formatting is finished, issue a ‘$PSPL,V=oldvol’ command: little by little, the old spool file will be drained (as job purges will occur) and eventually JES2 will unallocate this volume. You can also force JES2 to cancel all jobs that were using this volume ($PSPL,V=oldvol,CANCEL). Spool offload, followed by IPL, and spool reload is an alternative.
**JES2 checkpoint**

The checkpoint is crucial for JES2 processing. There is no need to perform a JES2 cold start – the checkpoint reconfiguration dialogue should be used to move it on the fly. Allocate a new checkpoint file, have JES2 know it ($TCKPTDEF,NEWCKPT1=) and start the migration process ($TCKPTDEF,RECONFIG=YES). You then reply with the appropriate FORWARD option. The process is very quick (it takes a bit longer if you have implemented the multi-access spool mode, because there will be additional replies). The recent versions of JES2 can dynamically allocate the space for the new checkpoint dataset. Don’t forget to update the CKPTDEF parameter in JES2PARM. The process is the same for the duplex checkpoint ($TCKPTDEF,NEWCKPT2=, etc).

**JES2 PROCLIBs**

If you want to take no chances, we would recommend that you copy JES2 proclibs without deleting the source file. If an IPL cannot be planned, an abnormal stop of JES2 can be forced ($PJES2,ABEND), followed by a JES2 hot start.

**JES3 spool**

Use ‘*F,Q,DD=(oldvol),DRAIN’ to drain the old volume. Unlike JES2, the new spool cannot be added on-the-fly. A JES3 warm start (or cold start) is mandatory to take the new spool file into account, whether it be formatted during JES3 initialization (FORMAT statement in the init deck) or pre-formatted (using the IEBDG utility to pad the file with X'FF's, and a TRACK statement in the init deck). Tape offload (dump job) can also be considered, because it is more handy than JES2 offload.

An alternative (which is rarely used, although IBM mentions it in the documentation) is to hot start JES3 without the volume to be moved. After moving the data to the new DASD volume (DFDSS full copy), a hot start with the dataset (on the new volume) allocated will oblige JES3 to consider the dataset available again.

**Other JES3 datasets**

Moving the JCT dataset (a file containing information about the status of jobs) normally requires a JES3 cold start. IBM also provides a
'JCTTOOL’ facility to help you. The checkpoint dataset is less of a problem, because it is used only to avoid reprocessing of the JES3 init deck each time JES3 is hot-started.

**SYS1.UADS**

DFDSS copy is sufficient for copying SYS1.UADS. From my point of view, the IBM utility UADSREFM is inferior because it does not copy entries for logged on users. You should use the TSO/E SYNC command to synchronize it with the RACF database and SYS1.BRODCAST. Recatalog the new SYS1.UADS and IPL your system.

A warning: never use IEBCOPY to move an UADS. If you do so, after IPLing with this new UADS, no user will be able to log on. This is because BDAM is used for accessing the UADS, not BPAM.

**SYS1.BRODCAST**

There is no problem with SYS1.BRODCAST: allocate a new file on the target volume and re-SYNC it. Then recatalogue it. An IPL is not required.

**SMF files**

Identify SMF files not currently in use (with the ‘D SMF’ command), suppress any reference to them in SMFPRMxx, issue the ‘SET SMF=xx’ command to get them unallocated, recreate them (DELETE, DEFINE CLUSTER with the MODEL subparameter). Re-update SMFPRMxx and reissue the ‘SET SMF=xx’ command for SMF to format them. No IPL is required except if you decide to change their CI size (as all must have the same CI size).

**HSM ML/1 volumes**

Never use DFDSS to move files between ML/1 volumes. Migrated files are not catalogued, they are referenced in HSM control datasets, so you must use HSM commands to move them.

- Prerequisite: the daily autobackup should have been run (verify that there is no HSM.HBACK file on the volume). If it is not
case, issue the FREEVOL ML1 BACKUPVERSIONS command. Also verify that there is enough room on other ML/1 volumes, and sufficient SDSP files (if you use small dataset packing).

- Drain the volume: ADDVOL volser UNIT(3390) MIG(ML1 DRAIN).
- Move the files: FREEVOL MVOL(volser) TARGETLEVEL(ML1) (the TARGETLEVEL parameter prevents tapes being mounted to receive files you want to move).
- Unassign the volume from HSM: DELVOL volser MIGRATION(PURGE).
- HSM.HMIG files remaining on the volume often result from past errors (HSM being cancelled, job or system crash, etc). You may DELETE NOSCRA TCH the related files (that appear as migrated, but are not known by HSM), whose name you’ll find by simply browsing the HSM.HMIG files. You can finally purge these HSM.HMIG files. Any non-HSM user datasets allocated on ML/1 volume should be moved in the normal way.

HSM back-up volumes
For HSM back-up volumes, the processing is the same as above, the commands to be used are:

FREEVOL BVOL(volser) TARGETLEVEL(SPILL(DASD)) (or: SPILL(TAPE))

DELVOL volser BACKUP(PURGE)

HSM control files (MCDS, BCDS, OCDS)
HSM control files are normal VSAM files. Stop HSM and invoke DFDSS to move them. The SDSP files (HSM.SMALLDS.Vxxxxxx) must not be moved, but can be deleted after a successful ML/1 volume migration.

Public work volumes
No real data migration is normally required for public work volumes, only MOUNT commands with the PRIVATE attribute in place of the PUBLIC one.
**DB2 or IMS databases**

There is no problem using DFDSS with DB2 or IMS databases (never use IDCAMS to move such files). Stop DB2 or IMS, or, alternatively stop the files you want to move using the commands:

- `-STOP DATABASE(database) SPACENAM(tablespace) for DB2.`
- `/DBR DB(database) NOFEOV for IMS (/DBR AREA for DEDB bases).`
- `If the IMS databases are used by CICS systems (local DL/I), issue a CEMT SET DLIDATABASE(database) STOP command (supposing the base was allocated through a DFSMDA macro).`

For DB2, ensure that volumes belong to the right STOGROUP. To change volume affinity, use the ALTER command (ALTER STOGROUP ... ADD VOLUMES(...) REMOVE VOLUMES(...)).

For IMS, in some rare cases (3380 to 3390 migration and KSDS defined with the BUFFERSPACE parameter), the index CI size of a HIDAM or HISAM database may change, which may affect the DFSVSMxx parameters.

**CICS user files**

If CICS user files were dynamically allocated (no DD card), issue a CEMT SET DATASET(database) CLOSE command to free them.

**IMS CONTROL DATASETS**

IMS control datasets apply to special cases, especially in a 3380/3390 migration. The following are examples:

- `ACBLIB: run an ACBGEN (BUILD PSB=ALL).`
- `WADS, RDS, message queues, SPA dataset: /NRE with FORMAT option recommended.`

Some IMS commands allow you to add a new WADS (/START WADS) or OLDS (/START OLDS) file. These files must be previously defined and recognized by IMS (DFSMDA macros for allocation, WADS or OLDS number falling in the defined range).
HFS (Hierarchical File System) files

HFS files are special SMS files for OpenEdition (the MVS flavour of Unix). They cannot be logically moved by the COPY command with the current versions of DFDSS, you must use the DUMP and RESTORE functions.

Page datasets

Use the ‘PAGEDELETE, PAGE=dsn’ to drain local page datasets.

PLPA and COMMON datasets cannot be moved. Note that there is no ENQ protecting page datasets, there is only a flag (UCBPGFL) in the UCB. Allocate new page datasets to replace them on the next IPL and update the IEASYSxx member in your PARMLIB. The procedure is the same for swap datasets.

Master catalog

The master catalog cannot be moved. AMS REPRO must be used to copy its contents to a new master catalog. To do this:

- Define the new master catalog (DEF UCAT)
- Repro the old to the new:
  ```
  //COPYCAT EXEC PGM=IDCAMS
  //STEPCAT DD DISP=SHR,DSN=CATALOG.NEWMCAT
  //     DD DISP=SHR,DSN=CATALOG.OLDMCAT
  //SYSPRINT DD SYSOUT=*
  //SYSSIN DD *
  REPRO IDS(CATALOG.OLDMCAT) ODS(CATALOG.NEWMCAT)
  ```
- Connect the old one to the new one (useful to delete the old after IPL):
  ```
  IMP OBJ((CATALOG.OLDMCAT -
     VOL(oldvol) DEVT(3390))) CON CAT(CATALOG.NEWMCAT)
  ```
- Update SYSCATxx or LOADxx, IPL, and finally, DEL CATALOG.OLDMCAT UCAT RECOVERY.

SYS1.LOGREC

Allocate a new LOGREC and recatalogue it. Use IFCDIP00 to format the new LOGREC. An IPL is required.
SYS1.STGINDEX

Who still uses Checkpoint/Restart? STGINDEX is an optional file, and the only risk with it missing is to receive a message during IPL:

ILR023I DYNAMIC ALLOCATION OF VIO JOURNALING DATA SET FAILED. NO VIO JOURNALING

The VIODSN parameter (in IEASYSxx) enables you to assign another name for the STGINDEX file, or, preferably, to avoid using it (VIDSN=IGNORE).

IODF files, SYSn.IPLPARM

There is no problem moving the IODF or SYSn.IPLPARM files, but do not forget they must reside on the volume pointed to by the first 4 bytes of the loadparm parameter (when IPLing the system, do not use the master catalog to locate them).

User catalogs

User catalogs are not the easiest part of file migration! They should be moved only when quiesced. They should not be moved together with the data sets that are catalogued in them.

Before the move, the catalog should be quiesced and freed on any other system, through the commands ‘F CATALOG,CLOSE (CATALOG.USER)’ and ‘F CATALOG,UNALLOCATE (CATALOG.USER)’. You can move catalogs that are not closed, the risk being that you receive some time later error messages when VSAM files are closed (eg when CICS is stopping), because VSAM tries to write statistics onto the catalog at its previous location.

Locking the catalog prevents users from opening it. To do so, an IGG.CATLOCK profile must exist in the RACF FACILITY resource class. This is a JCL to achieve the move:

```
//LOCKCAT EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=* 
//SYSIN DD *
ALTER CATALOG.USER LOCK /* prevents any access */
/**
//SAVECAT EXEC PGM=ADRDSSU save the usercat in case...
//SYSPRINT DD SYSOUT=* 
//SAVECAT DD UNIT=SYSALLDA,DISP=(NEW,CATLG),DSN=xxx.SAVECAT,
// SPACE=(CYL,(9,5),RLSE)
//SYSIN DD *
```
Saving the catalog before the move is a good precaution, because the move step can be cancelled, or be waiting forever, there is a risk of getting a damaged catalog.

A less interesting alternative is to REPRO MERGECAT the source catalog to a new target one. You must afterwards rectify all the related aliases. If the catalog is shared between several systems, you must reconnect it correctly to any other master catalog:

```
IMPORT OBJ((CATALOG.USER VOL(newvol) DEVT(3390))) -
CONNECT ALIAS CAT(CATALOG.MASTER)
```

If the foreign system has not taken into account the new location for the catalog, you may issue a F CATALOG,RESTART command to force it to do so.

One aspect to consider: do not EXPORT DISCONNECT the catalog before the IMPORT CONNECT (as you did in the old way of moving a catalog, before IBM invented IMPORTCONNECT ALIAS), because you would lose all the aliases related to the catalog.

**SMS control datasets (ACDS, COMMDS)**

You cannot stop the SMS address space, but the SETSMS command enables you to use different control datasets. To assign a new ACDS, define it and issue the commands:

- SETSMS SAVEACDS(new-acds) – to save the current SMS configuration.
- SETSMS ACDS(new-acds) – to enable the new ACDS.

To assign a new COMMDS use ‘SETSMS COMMDS(new-commds)’. Adapt the IGDSMSxx member of your PARMLIB.
Linklist libraries

Use DFDSS to copy linklist libraries (no DELETE, use the TOL(ENQF) parameter), re-catalogue them on the target volume, and IPL. If you really want to move them on-the-fly, use some utilities like RESOLVE (from Boole and Babbage) or LNKLST statements in PROGxx (beginning with OS/390 Release 3).

Some libraries are always allocated and will never be moved (linklist libraries, VTAMLIB, TSO libraries, etc), but will be copied and recatalogued. It’s a pity that DFDSS does not even allow you to uncatalog/recatalog files that are in use (DFDSS requires exclusive use to do so).

SYS1.DUMPxx

Dump datasets can be unallocated by a DD DEL,DSN=(xx) command, and reallocated by DD ADD,DSN=(xx).

SYS1.DAE

You must disable DAE (Dump Analysis and Elimination). Usually IBM provides an ADYSET01 member in PARMLIB. Try the SET DAE=01 command.

SYS1.PAGEDUMP

SYS1.PAGEDUMP is the stand-alone dump program dataset. This file must reside on the S.A. volume. No problem if you lose this file (except that you can no longer invoke the S.A. dump facility). You should regenerate the S.A. dump program.

RACF databases

RACF databases can be migrated on-the-fly if a back-up database is in use and the RACF activity is low. A job can be written to automate the process:

- RVARY SWITCH – the primary RACF base is deallocated and replaced by the back-up one.
- Copy the primary database to a new primary (ICHUT400 or IRRUT400).
• I would then run an ICHUT200 or IRRUT200 step to validate the structure of the new RACF database.
• Catalogue the new primary base and uncatalogue the old one.
• RVARY SWITCH – the new primary RACF database is activated and the back-up one is deallocated.
• RVARY ACTIVE DATASET(backupbase) – the backup RACF database is activated again.

The RVARY command must be synchronized on all systems if the RACF database is shared. The procedure is similar for back-up datasets (you copy the primary on the new backup file).

CICS journals
CICS journals are sensitive to DASD characteristics. For a 3380/3390 migration, allocate new journals, format them with the DFHJJCJFP utility, then cold start CICS.

THE MOST FREQUENT PROBLEMS WITH DASD MIGRATION
These are actual cases that have been encountered in practice:

1. VSAM files with an expiration date (due to DEFINE CLUSTER issued with subparameters FOR(days) or TO(date)).

   This is a minor problem because it provokes only post-processing errors. These files are correctly copied to the target volume, the source files are renamed by DFDSS but not deleted (you receive a code 8 with the message CATALOG ERROR WHEN DELETING CLUSTER). You have to purge them manually (DELETE xxx PURGE). If you spot them beforehand, you should issue the command ALTER xxx NULLIFY(RETENIóN) for each of them before attempting to move them.

   The same problem exists with non-VSAM files with an expiration date (the consequence of it is the source file will remain on the source volume, with the same name).

2. Files moved even if they are allocated to other users.
Normally, you cannot move files that are in use, but some products allocate files with no enqueue (option NO DATASET INTEGRITY in the MVS program property table). You may move these files to other disks without any problem, except that those products could receive some abends due to the fact that their files are no longer on the volumes they thought they were. Known examples are: PSF, JES2, JES3 (did you ever try to rename/compress/delete a JES PROCLIB ?...).

The same problem occurs if files are allocated by tasks on a foreign system and you have no inter-system integrity tool (GRS, MIM).

3 Trying to move files from an unrecognized SMS volume.

The SMS volume must be known by your system (it must belong to an active storage group in your system) and must not be disabled. DFDSS will not proceed (even if you try a full volume copy).

4 Volumes referenced in user JCLs (VOL=SER=xxx, DEF CL(... VOLUME(xxx)), etc) or even in programs (dynamic allocation).

There is no way other than changing all references (IPOUPDTE may be sufficient for that), except if the volumes are SMS managed (you should only define the old volume in a dummy storage group, and let SMS rules guide new allocations). The same problem occurs with UNIT=3380 in JCL if you are migrating to 3390.

5 Running physical and logical copies together.

Be cautious when physical and logical volume copies are executing in parallel. For example, if a volume containing a user catalog is physically copied and put off-line while some logical copies are running, the catalog could become corrupted or contain incorrect entries for files that were moved by logical copies.

6 Reinitializing a shared disk.

The first thing to do is to set the address off-line on all systems sharing the disk. Initializing a disk on one system while it is on-line on another can provoke allocation problems.
The next pitfalls are very common (from my own experience) they all stem from catalog problems. Catalog management in MVS is so flexible that there is always a risk of missing something, due to what I call hidden objects.

7 SMS managed files not catalogued.

This should normally not exist. It is a minor problem, as nobody can use these files anyway. You should spot them beforehand with a job like this:

```
//STEP1 EXEC PGM=ADRDSSU,PARM='TYPRUN=NORUN'
//SYSPRINT DD SYSOUT=*
//TAPE DD DUMMY
//SYSIN DD *

DUMP DS(INC(**),BY((CATLG,EQ,NO)), -
    EXCL(SYS1.VTOCIX.*,SYS%%%%%.T*.**)) OUTDD(TAPE) -
    INDD(DD1,DD2,DD3,DD4,DD5,DD6,DD7,DD8,DD9...)
//DD1 DD UNIT=SYSALLDA,DISP=SHR,VOL=SER=sms001
//DD2 DD UNIT=SYSALLDA,DISP=SHR,VOL=SER=sms002
```

The DDx cards all point to your SMS volumes. These files should be deleted (if they do not belong to a foreign SMS complex!) with commands like DELETE xxx FILE(DD1) NVR, or, on the contrary, be recatalogued (DEF NVSAM ... RECATALOG).

Same problem with VSAM files not catalogued – not a rare event in many data centres (try DELETE xxx VVR).

8 Moving a file catalogued in a foreign master catalog.

A classic problem with systems sharing disks. The foreign system will not be informed that its files have been moved (so IPLing this system can generate nasty surprises). You must recatalogue these files properly. Find them in advance with a LISTCAT ALL command.

9 The hidden catalog entry – moving a file catalogued both in your master catalog and in a user catalog.

Yes, you may have both a ‘normal’ alias called, say, PROD, and a PROD.LIBRARY catalogued in the master catalog (try it). This ‘double cataloguing’ is a trick used sometimes by systems people, often for linklist or IPA-list libraries. The entry in the master catalog is used during the IPL phase, while the one in the user catalog will be used the rest of the time. The problem is that
DFDSS, when moving these files, will update the user catalog, not the master catalog (hence problems again when IPLing). Look for these files using a LISTCAT ALL towards your master catalog. The recent versions of ISPF (DSLIST, option 3.4) now display these double entries. Prepare in advance DEL xxx NOSCRATCH CAT(mastercat) statements, and DEF NVSAM ... CAT(mastercat) statements to take the new location into account.

10 The hidden alias – lost when moving.

This is when, for example, your current DFSORT library is called PROD.SORTLIB (this name is used in all your JCL) but its real name (on the disk) is SYS1.SORTLIB. When you move it, DFDSS will recatalogue it but the alias PROD.SORTLIB will be deleted (without any warning). You must recreate it with a command like: ‘DEF ALIAS(NAME(PROD.SORTLIB) REL(SYS1.SORTLIB))’. These entries can be found in advance with a LISTCATALIAS command directed to all your user catalogs.

From what we could test, the problem does not exist with VSAM files (PATH entries related to primary clusters are preserved when the clusters are moved).

11 The hidden candidate – premature reinitializing of a candidate volume.

You were happy to have emptied many volumes containing VSAM files, and eventually you re-initialized them (ICKDSF INIT). Now, some people are reporting errors when deleting VSAM files, or when the system tries to expand a VSAM file onto another volume. This was because the volumes you had reinitialized were candidate volumes for these files (though no space was allocated on them). An ALTER xxx REMOVEVOLUMES(volser) for any concerned file will fix the problem.

The same problem exists for non VSAM files with more annoying consequences: as soon as people try to handle or delete these not really multi-volume files, they get the message:

IEF238D - REPLY DEVICE NAME OR 'CANCEL'.

because, as the system is unable to know whether or not the
candidate volume was used for these files, it tries to inspect the newly disappeared volume.

You can be proactive by listing all the catalog entries and searching for occurrences of these volumes, that you want to reinitialize, as current candidates for VSAM or non-VSAM files.

CONCLUSIONS

To conclude, disk migration is often a tedious task. Take no chances! Try to visualize all the problems and all the special cases. Prepare all the JCL, before the migration day, and test it carefully. There will be enough new problems you had not foreseen.

Thierry Fallissard
Technical director
etic (France) © Xephon 1998

Year 2000 aid: replace source strings – part 2

This month we complete our look at the year 2000 utility to replace source strings.

GETREC   ST    RBAL,SAVGRBAL  SAVE LINKAGE REGISTER
          L     R1,INRECLOC     POINT TO RECORD LOCATION
          LTR   R1,R1           FIRST RECORD OF MEMBER?
          BNZ   GRNXTREC        NO
          GRNXTBLK LA    R2,DECBA    POINT TO DECB
          L     R3,BLOCKLOC     POINT TO AREA ADDRESS
          ST    R3,INRECLOC     SAVE RECORD POINTER
          READ  (R2),SF,PDS,(R3),MF=E  READ BLOCK FROM MEMBER
          CHECK (R2)             AWAIT ECB POSTING
          LH    R5,INLRECL      LOAD RECORD LENGTH
          LH    R3,INBLKSIZ     LOAD MAXIMUM BLOCK SIZE
          L     R1,DECBA+16     LOAD RECORD POINTER WORD (IOB)
          SH    R3,14(R1)       SUBTRACT REMAINING COUNT
          L     R1,BLOCKLOC     GET ADDRESS OF BLOCK
          AR    R3,R1           POINT TO END OF BLOCK
          BCTR  R3,Ø            POINT TO LAST BYTE OF BLOCK
ST R3,BLOCKEND  SAVE ENDING ADDRESS
L R1,INRECLOC  POINT TO BEGINNING OF BLOCK
B GRISTREC  GO PROCESS FIRST RECORD OF BLOCK
GRNXTREC L R1,INRECLOC  GET PREVIOUS RECORD LOCATION
AH R1,INLRECL  POINT TO NEXT RECORD
C R1,BLOCKEND  PAST END OF BLOCK?
BNL GRNXTBLK  YES
GRISTREC ST R1,INRECLOC  SAVE ADDRESS OF RECORD
XR R15,R15  SET 'RECORD FOUND' CODE
AP RECORDS,=P'1'  COUNT RECORD
GRRETURN L RBAL,SAVGRBAL  RESTORE LINKAGE REGISTER
BR RBAL  RETURN
GREOF LA R15,4  SET 'RECORD NOT FOUND' CODE (EOF)
BR GRRETURN  GO RETURN
EJECT

***********************************************************************
***   GET PDS ISPF STATISTICS                                       ***
***********************************************************************
GETSTATS ST RBAL,SAVGBAL  SAVE LINKAGE REGISTER
XC BLDLNTRY(BLDLLLEN),BLDLNTRY  CLEAR ENTRY WORK AREA
MVI GUØ2FF+1,X'Ø1'  SET ENTRY COUNT TO 1
MVI GUØ2LL+1,X'5Ø'  SET ENTRY LENGTH TO 8Ø
MVC GUØ2NAM,MEMBER  MOVE MEMBER NAME INTO BLDL AREA
LA R1,PDS  R1 POINTS TO OPEN DCB
LA RØ,BLDLNTRY  RØ POINTS TO BLDL ENTRY AREA
BLDL (R1),(RØ)  EXECUTE BLDL
LTR R15,R15  TEST RETURN CODE
BZ GSRETURN  IF NOT NORMAL RETURN
TM GUØ2C,X'8Ø'  IF AN ALIAS
BNO GSRETURN  THEN
LA R15,12  TURN ON ALIAS FLAG
BS GETSTATS  EXIT IF NOT NORMAL RETURN
END GETSTATS

***********************************************************************
***   WRITE TSO STATISTICS                                          ***
***********************************************************************
PUTSTATS ST RBAL,SAVPSBAL  SAVE LINKAGE REGISTER
OC GUØ2DATC,GUØ2DATC  CREATION DATE BINARY ZEROS?
BZ RDYNOSTAT  YES
MVC LINE+1(11),=C'ISPF STATS:,'
UNPK LINE+13(6),GUØ2TTR(L'GUØ2TTR+1) UNPACK TTR NYBLS
NC LINE+13(5),=8X'F'  MASK OUT ZONES
TR LINE+13(5),=C'Ø123456789ABCDEF' CONVERT TO DIXPLAY
XR R1,R1  CLEAR REGISTER
IC R1,GUØ2MOD  GET MODIFICATION
ST R1,DOUBLE  SAVE
EJECT
IC R1,GUØ2VER GET VERSION
MH R1,-H'100' MOVE 2 DECIMAL DIGITS LEFT
A R1,DOUBLE ADD MODIFICATION
CVD R1,DOUBLE CONVERT TO DECIMAL
MVC LINE+18(7),=X'40202120B2020' SET EDIT PATTERN
ED LINE+18(7),DOUBLE+5 FORMAT VV.MM
ICM R1,B'1111',GUØ2DATC GET CREATION DATE
ST R1,JGYYDDD SAVE FOR CONVERSIONT
BAL RBAL,JULGREG CONVERT TO MM/DD/YY
MVC LINE+26(8),JGMMDDYY MOVE TO LINE
ICM R1,B'1111',GUØ2DATM GET CREATION DATE
ST R1,JGYYDDD SAVE FOR CONVERSIONT
BAL RBAL,JULGREG CONVERT TO MM/DD/YY
MVC LINE+35(8),JGMMDDYY MOVE TO LINE
UNPK LINE+46(5),GUØ2TIMM(3) UNPACK MODIFIED TIME
MVC LINE+45(2),LINE+46 MOVE HH LEFT
MVI LINE+47,C':' SEPARATE HH:MM
LH R1,GUØ2SIZE LOAD SIZE FROM DIRECTORY
CVD R1,DOUBLE CONVERT TO DECIMAL
MVC LINE+50(7),EDITPAT SET EDIT PATTERN
ED LINE+50(7),DOUBLE+5 FORMAT SIZE
LH R1,GUØ2INIT LOAD INITIAL SIZE FROM DIRECTORY
CVD R1,DOUBLE CONVERT TO DECIMAL
MVC LINE+57(7),EDITPAT SET EDIT PATTERN
ED LINE+57(7),DOUBLE+5 FORMAT SIZE
ICM R1,B'0011',GUØ2MOD LOAD COUNT OF MOD LINES
CVD R1,DOUBLE CONVERT TO DECIMAL
MVC LINE+64(7),EDITPAT SET EDIT PATTERN
ED LINE+64(7),DOUBLE+5 FORMAT SIZE
MVC LINE+71(7),GUØ2ID MOVE USER ID TO LINE
BAL RBAL,PRINT PRINT STATISTICS
L RBAL,SAVPSBAL RESTORE LINKAGE REGISTER
BR RBAL RETURN
EJECT

***********************************************************************
*** REWRITE ANY CHANGED RECORDS ***
***********************************************************************
WRITEREC ST RBAL,SAWWRBAL SAVE LINKAGE REGISTER
LA 2,DECBA POINT TO DECB
WRITE (2),SF,PDS,MF=E READ BLOCK FROM MEMBER
CHECK (2) AWAIT ECB POSTING
NI SWITCHES,X'FF'-UPDATBIT RESET UPDATE BIT
WRRETURN L RBAL,SAWWRBAL RESTORE LINKAGE REGISTER
BR RBAL RETURN
EJECT

***********************************************************************
*** WRITE ERROR LINES ***
***********************************************************************
PUTERR ST RBAL,SAVPEBAL SAVE LINKAGE REGISTER
AP ERRORTOT,=P'1' COUNT ERROR

© 1998. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (940) 455 7050, fax (940) 455 2492.
MVC INAREA+L'INAREA(9),=C'<=BEFORE' SET IMAGE
PUT ERRORS,OUTAREA WRITE BEFORE IMAGE
MVC INAREA+L'INAREA(9),=C'<=AFTER ' SET IMAGE
L R1,INRECLOC POINT TO MODIFIED RECORD
MVC INAREA,Ø(R1) MOVE AFTER IMAGE
PUT ERRORS,OUTAREA WRITE AFTER IMAGE
L RBAL,SAVPEBAL RESTORE LINKAGE REGISTER
BR RBAL RETURN
EJECT

***********************************************************************
*** THIS ROUTINE REPLACES THE STRING POINTED TO BY REGISTER R6 ***
*** WITH STRING DEFINED BY STDEF MACRO. ***
*** IF THE TARGET STRING IS NOT LONGER THAT THE OBJECT STRING ***
*** OR THERE IS SUFFICIENT SPACE IMMEDIATELY AFTER THE TARGET ***
*** STRING, REPLACEMENT IS DIRECT. ELSE, MULTIPLE CONSECUTIVE ***
*** SPACES ARE REMOVED FROM THE RECORD (AS NEEDED). ***
*** THE STDEF DEFINITION IS POINTED TO BY REGISTER R15. ***
***********************************************************************

REPLACE ST RBAL,SAVRPBAL SAVE LINKAGE REGISTER
STM R14,R5,SAVRETO5 SAVE WORK REGISTERS
NI SWITCCHES,X'FF'-ERRORBIT INITIALIZE ERROR FLAG (OFF)
BAL RBAL,FINDSP REDO SPACE ALLOCATION
TM OPTIONS,DIAGBIT DIAGNOSE?
BZ RPNOLIST NO
BAL RBAL,TESTX PRINT DIAGNOSTIC LINE
RPNOLIST XR R2,R2 CLEAR REGISTER
XR R3,R3
IC R2,Ø(R15) LOAD LENGTH OF STRING TO BE REPLACED
IC R3,1(R15) LOAD LENGTH OF REPLACEMENT STRING
LA R5,4(R2,R15) POINT TO REPLACEMENT STRING
MVC SAVEFLAG,2(R15) SAVE REPLACEMENT OPTION FLAG
CR R2,R3 COMPARE LENGTH OF TARGET TO OBJECT
BH RPTARGLT NO (OBJECT SHORTER)
BE RPTARGEQ NO (SAME SIZE)
SR R3,R2 COMPRESSION BYTES NEEDED
LA R4,AVSPI LOAD ADDRESS OF BLANK SEGMENT VECTOR
LA R6,AVS P2 LOAD ADDRESS OF END OF TABLE
LA R1,1(R6,R2) FIRST BYTE AFTER TARGET
RPISTLP C R1,Ø(R4) IMMEDIATELY TO THE RIGHT OF TARGET?
IGH
BE RPISTHIT YES
BL RPISTLPO NO, RIGHT OF TARGET
LA R4,L'AVSPI(R4) GET NEXT VECTOR ENTRY POSITION
CR R6,R4 PAST END?
BNL RPISTLP NO
B RPISTLP YES
RPISTHIT CH R3,4(R4) ROOM FOR REPLACEMENT STRING?
BNH R PTARGEQ YES

RPGTLPO LA   R2,AVSP1       LOAD ADDRESS OF BLANK SEGMENT VECTOR
RPGTLP1 LR   R4,R3        MAXIMUM BYTES TO COMPRESS
OC   4(R2),4(R2)       AVAILABLE SPACE?
B2   RPGTLP1X          NO
CH   R4,(R2)          SUFFICIENT SPACE FOR ENTIRE COMPRESS
BNH  RPGTLP1F         YES
LH   R4,(R2)          LOAD AVAILABLE SPACE
RPGTLP1F C   R6,0(R2)    IS SPACE BEFORE TARGET STRING?
BL   RPGTLPO          NO
LR   R0,R2           SAVE REGISTER
L    R2,0(R2)        LOAD ADDRESS OF 1ST BLANK
BAL  RBAL,MOVELEFT    MOVE DATA LEFT
LR   R2,R0           RESTORE REGISTER
B    RPGTLPA         ADJUST REMAINING LENGTH
RPGTLPA LR   R0,R2       SAVE REGISTER
L    R2,0(R2)        LOAD ADDRESS OF 1ST BLANK
LA   R1,1(R6,R7)     POINT TO BYTE AFTER TARGET
CR   R1,R2         IS THIS THE AVAILABLE SPACE?
BNE  RPGTLPO1       NO
MVI  Ø(R1),C'_'      SET TO NONBLANK (HIDE FOR NXT FINDSP
P)
RPGTLPO1 BAL  RBAL,MOVERIGHT   MOVE DATA RIGHT
LR   R2,R0           RESTORE REGISTER
RPGTLPA LH   R0,4(R2)     LOAD BYTES AVAILABLE
SR   R0,R4         REDECREMENT BY SIZE OF COMPRESSION
STH  R0,4(R2)       SAVE UPDATED LENGTH
SR   R3,R4         DEDUCT FROM TOTAL
BNP  RPTARGEO      EXIT IF COMPRESSION COMPLETE
BAL  RBAL,FINDSP    REDO SPACE ALLOCATION
B    RPGTLPO       GO RESTART FROM BEGINNING
RPGTLPX LA   R2,LA'AVSP1(R2)  POINT TO NEXT VECTOR ELEMENT
LA   R0,AVSP2      POINT TO LAST ELEMENT
CR   R2,R0        PAST END OF TABLE?
BNH  RPGTLP1      NO
OI   SWITCHES,ERRORBIT  SET ERROR FLAG
TM   SAVEFLAG,X'80'    IS OPTION=FORCE?
B0   RPFITOK      NO
L    R2,INRECLOC    POINT TO CURRENT RECORD
LA   R2,71(R2)      POINT TO END OF RECORD
LR   R4,R3        NUMBER OF UNPROCESSED BYTES
BAL  RBAL,MOVERIGHT OVERLAY RIGHT MOST CHARACTERS
B    RPTARGEO     GO REPLACE
RPTARGEO LA   R1,0(R6,R2)     END OF ORIGINAL STRING
CLI  1(R1),C' '     IS NEXT CHARACTER BLANK?
BNE  RPTARGED     NO
MVI Ø(R6),C' '     CLEAR 1ST BYTE OF ORIGINAL STRING
EX   R2,RPADRT    CLEAR REMAINING BYTES
B    RPTARGEO     GO OVERLAY TARGET STRING
RPTARGED LR   R4,R2      LENGTH OF EXISTING STRING
SR   R4,R3       LESS LENGTH OF REPLACEMENT STRING
*** CONVERSION ROUTINE ***

**CONVERT JULIAN DATE TO GREGORIAN DATE**

```
JULGREG ST RBAL,SAVGJ.Globalization REG T SAVE LINKAGE REGISTER
CLI JGYYDDD,1 IS ACTUAL CENTURY PRESENT?
BH JGFACTUAL YES
TR JGYYDDD(1),=X'1920' CENTURY=Ø ==> 19XX, 1==>20XX
JGFACTUAL ZAP JGDAYS,JGYYDDD+2(2) SAVE DAYS FROM BEGINNING OF YEAR
ZAP JGMONTHS,=P'1' INITIALIZE MONTH
LA R15,JANUARY POINT TO FIRST MONTH OF YEAR
LA RØ,L'JANUARY SIZE OF DAYS/MONTH FIELD
LA R1,DECEMBER POINT TO LAST MONTH OF YEAR
ZAP FEBRUARY,=P'28' SET NON LEAP YEAR DAYS
CLC =X'2000',JGYYDDD YEAR 2ØXX?
BE JGYR2ØØØ YES
JG2ØTHCN TM JGYYDDD+1,1 LEAP YEAR?
BO JGLOOP NO
TM JGYYDDD+1,X'12'
BM JGLOOP NO
JGYR2ØØØ AP FEBRUARY,=P'1' ADJUST
JGLOOP CP JGDAYS,Ø(L'JANUARY,R15) CURRENT MONTH?
BNH JGFOUND YES
AP JGMONTHS,=P'1' INCREMENT MONTH
SP JGDAYS,Ø(L'JANUARY,R15) DECREMENT DAYS PER CURRENT MONTH
BXLE R15,RØ,JGLOOP CONTINUE
JGFOUND UNPK JGMMDDYY(2),JGMONTHS UNPACK MONTH
UNPK JGMMDDYY+3(2),JGDAYS UNPACK DAY
UNPK JGMMDDYY+6(3),JGYYDDD(2) UNPACK YEAR
MVI JGMMDDYY+2,C'/' SEPARATE MONTH AND DAY
MVI JGMMDDYY+5,C'/' SEPARATE DAY AND YEAR
OI JGMMDDYY+1,C'Ø' FORCE MONTH NUMERIC
OI JGMMDDYY+4,C'Ø' FORCE DAY NUMERIC
OI JGMMDDYY+7,C'Ø' FORCE YEAR NUMERIC
UNPK JGMDDC+6(5),JGYYDDD(3) UNPACK CCYY
MVC JGMDCC+6(6),JGMMDDYY SET MM/DD/
JGRETURN L RBAL,SAVGJ.Globalization REG T LOAD LINKAGE REGISTER
BR RBAL RETURN
EJECT
```

**PRINT ROUTINE**

```
PRINT PUT PRINTER,LINE PRINT LINE
MVI LINE,C' ' SET SEED
MVC LINE+1(L'LINE),LINE CLEAR LINE
DOUBLESP BCTR R9,RBAL RETURN IF PAGE NOT FULL
HEADPAGE MVC PAGENO,=X'4Ø2Ø212Ø' SET EDIT PATTERN
ED PAGENO,PAGES FORMAT PAGE NUMBER
AP PAGES,=P'1' INCREMENT PAGE COUNT
PUT PRINTER,HEADER PRINT PAGE HEADING
```
*** THIS ROUTINE PERFORMS A SCAN OF STATEMENTS TO DETERMINE ***
*** POTENTIAL COMPRESSION POINTS. IE TWO OR MORE SPACES ***
*** NOT CONTAINED IN SINGLE QUOTES ***
***********************************************************************
FINDSP  ST    RBAL,SAVFSBAL       SAVE LINKAGE REGISTER
STM   RØ,R6,FSREGSAV      SAVE WORKING REGISTERS
XC    AVSP1(AVSP2-AVSP1+L'AVSP1),AVSP1 CLEAR VECTOR
L     R6,INRECLOC         POINT TO BEGINNING OF RECORD
LA    R8,71(R6)           POINT TO END OF RECORD
LA    R6,1(R6)            DON'T OVERLAY COLUMN 1
LA    R3,AVSP1            POINT TO FIRST AVAILABLE SPACE
LA    R4,AVSP2            POINT TO LAST AVAILABLE SPACE
FSLOOP  XC    TRTAB,TRTAB         CLEAR TRANSLATION TABLE
MVI   TRTAB+C' ',C' '     TURN ON BLANK POSITION
MVI   TRTAB+C'''',C''''   TURN ON QUOTE POSITION
LR    R2,R8               POINT TO LAST LOCATION
SR    R2,R6               MAXIMUM LENGTH OF SCAN
BNP   FSRETURN            END IF NOT MORE THAN 1 POSITION
EX    R2,TRT              SCAN FOR QUOTE OR SPACE
BZ    FSRETURN            EXIT IF NONE
LA    R6,1(R1)            POINT PAST OCCURRENCE
LR    R2,R8               POINT TO LAST POSITION
SR    R2,R6               SUBTRACT CURRENT POSITION
BNP   FSRETURN             END IF NOT MORE THAN ONE SPACE
CLI  Ø(R1),C''''         QUOTE?
BE    FSQUOTE              YES
LR    RØ,R1               SAVE POSITION
LA    R1,1(R8)            SET DEFAULT LOCATION FOR NULL
MVC   TRTAB,TRTAB-1       SET ALL TO NONZERO
MVI   TRTAB+C''',Ø        SET BLANK POSITION TO ZERO
EX    R2,TRT              SCAN FOR FIRST NON-BLANK
SR    R1,R6               CALCULATE NUMBER OF BLANKS
AR    R6,R1               POINT TO NON-BLANK
CH    R1,=H'1'            AT LEAST 2 CONSECUTIVE BLANKS?
BL    FSLOOP               NO
ST    RØ,Ø(R3)            SAVE LOCATION OF SPACES (1ST BYTE)
STH   R1,4(R3)            SAVE NUMBER OF SPACES
LA    R3,6(R3)            POINT TO NEXT LOCATION/LENGTH PAIR
CR    R3,R4               IS VECTOR FULL?
BH    FSRETURN             YES
B    FSLOOP              GO CONTINUE
FSQUOTE  MVI   TRTAB+C' ',Ø        TURN OFF SPACE POSITION
EX    R2,TRT              SCAN FOR NEXT QUOTE
BZ    FSRETURN             EXIT IF NON FOUND
LA    R6,1(R1)            POINT PAST QUOTE
B    FSLOOP  CONTINUE
FSRETURN LM  R0,R6,FSREGSAV  RESTORE WORKING REGISTERS
L    RBAL,SAVFSBAL  RESTORE LINKAGE REGISTER
BR    RBAL  RETURN
EJECT
*******************************************************************************
***   SCAN FOR IMBEDDED STRINGS                                           ***
*******************************************************************************

SCAN1    ST    RBAL,SAVS1BAL  SAVE LINKAGE REGISTER
XR    R3,R3  CLEAR REGISTER
L    R6,INRECLOC  LOAD ADDRESS OF INPUT RECORD
LA    R8,72  NUMBER OF BYTES
LR    R5,R8  FOR LENGTH-1 COMPARISON
S1LOOP2  LA    R15,WORDLIST  POINT TO LIST OF STRINGS
LA    R14,TOTALS  POINT TO ACCUMULATORS FOR 1ST STRING
BCTR  R5,Ø  REMAINING LENGTH - 1
S1LOOP1  IC    R3,Ø(R15)  INSERT LENGTH-1 OF WORDLIST STRING
TM    2(R15),WORDBIT+PREFBIT+SUFXBIT  IMBEDDED STRING VALID?
BNZ   S1LOOP1X  NO
CR    R5,R3  PAST END OF INPUT?
BL    S1LOOP1X  YES
EX    R3,S1CLC  MATCH FOUND?
BNE   S1LOOP1X  NO
XR    R7,R7  CLEAR REGISTER
IC    R7,Ø(R15)  SET TARGET LENGTH
BAL   RBAL,REPLACE  GO ATTEMPT TO REPLACE STRING
AP    Ø(L'TOTALS,R14),=P'1'  COUNT OCCURRENCE
MVI   HIT,X'FF'  FLAG RECORD
LA    R0,Ø(R6,R7)  POINT TO END OF REPLACEMENT STRING
SR    R0,R6  NEW LENGTH - 1
AR    R6,R0  ADJUST POINTER
SR    R8,R0  ADJUST LENGTH
BNP   S1RETURN  EXIT IF END OF STATEMENT
S1LOOP1X XR    R0,R0  CLEAR REGISTER
IC    R0,1(R15)  INSERT LENGTH OF REPLACEMENT
LA    R15,5(R3,R15)  POINT TO REPLACEMENT STRING 2ND CHAR
AR    R15,R0  POINT TO NEXT WORDLIST PAIR
LA    R14,8*L'TOTALS(R14)  POINT TO CORRESPONDING TOTALS
CLI   Ø(R15),X'FF'  END OF LIST?
BNE   S1LOOP1  NO
LA    R6,1(R6)  POINT TO NEXT CHARACTER
BCT   R8,S1LOOP2  CONTINUE
S1RETURN L    RBAL,SAVS1BAL  RESTORE LINKAGE REGISTER
BR    RBAL  RETURN
S1CLC  CLC   3(*-*,R15),Ø(R6)
EJECT
*******************************************************************************
***   SCAN FOR WORDS, PREFIXES, AND SUFFIXES                               ***
*******************************************************************************

SCAN2    ST    RBAL,SAVS2BAL  SAVE LINKAGE REGISTER
XR R3,R3
CLEAR REGISTER
L R6,INRECLOC LOAD ADDRESS OF INPUT RECORD
BCTR R6,Ø DECREMENT TO PREVIOUS BYTE
LA R8,72 NUMBER OF BYTES
XR R7,R7 INITIALIZE LENGTH
S2LOOP2 LA R15,WORDLIST POINT TO LIST OF STRINGS
LA R14,TOTALS POINT TO ACCUMULATORS FOR 1ST STRING
LTR R7,R7 IS LENGTH NEGATIVE?
BM S2RETURN YES (SHOULDN'T HAPPEN)
BAL RBAL,GETWORD SCAN FOR VALID STRING
LTR R8,R8 RECORD DEPLETED?
BNP S2RETURN YES
S2LOOP1 IC R3,(R15) INSERT LENGTH-1 OF STRING 2B RPLCD
CR R7,R3 PAST END OF INPUT?
BL S2LOOP1X YES
BNE S2NOTW CAN'T BE WORD MATCH UNLESS SAME SIZE
TM 2(R15),WORDBIT WORD COMPARISON?
BZ S2NOTW NO
EX R3,S1CLC MATCH FOUND?
BNE S2RETURN NO
AP L'TOTALS(L'TOTALS,R14),=P'1' COUNT OCCURRENCE
B S2FOUND GO FLAG RECORD
S2NOTW TM 2(R15),PREFBIT PREFIX COMPARISON?
BZ S2NOTP NO
EX R3,S1CLC MATCH FOUND?
BNE S2RETURN NO
AP 2*L'TOTALS(L'TOTALS,R14),=P'1' COUNT OCCURRENCE
B S2FOUND GO FLAG RECORD
S2NOTP TM 2(R15),SUFXBIT SUFFIX COMPARISON?
BZ S2LOOP1X NO
LA R1,(R6,R7) POINT TO END OF INPUT STRING
SR R1,R3 LESS LENGTH OF WORDLIST SUFFIX
EX R3,S2CLC MATCH FOUND?
BNE S2LOOP1X NO
AP 3*L'TOTALS(L'TOTALS,R14),=P'1' COUNT OCCURRENCE
S2FOUND MVI HIT,X'FF' FLAG RECORD
BAL RBAL,REPLACE GO ATTEMPT TO REPLACE STRING
S2LOOP1X XR R0,R0 CLEAR REGISTER
IC R0,1(R15) INSERT LENGTH OF REPLACEMENT
LA R15,5(R3,R15) POINT TO REPLACEMENT STRING 2ND CHAR
AR R15,R0 POINT TO NEXT WORDLIST PAIR
LA R14,8*L'TOTALS(R14) POINT TO CORRESPONDING TOTALS
CLI Ø(R15),X'FF' END OF LIST?
BNE S2LOOP1 NO
B S2LOOP2 CONTINUE
S2RETURN L RBAL,SAVS2BAL RESTORE LINKAGE REGISTER
BR RBAL RETURN
S2CLC CLC 3(*-*),(R15),Ø(R1) EJECT
***********************************************************************
*** SCAN FOR ALPHAMERIC STRING ***

******************************************************************************
GETWORD ST RBAL,SAVGWBAL SAVE LINKAGE REGISTER
LA R6,1(R6,R7) POINT PAST CURRENT STRING
SR R8,R7 SUBTRACT LENGTH-1 OF PREVIOUS STRING
BCTR R8,0 " OTHER BYTE
LTR R8,R8 ANY REMAINING DATA?
BNP GWRETURN NO
LA R1,0(R6,R8) POINT TO END OF TEXT
EX R8,GWTRT1 FIND FIRST NON-BLANK/SPECIAL
BZ GWNULL EXIT IF NONE FOUND
LR R7,R1 GET STARTING ADDRESS OF STRING
SR R7,R6 COMPUTE LENGTH-1 OF EMPTY SPACE
SR R8,R7 REDUCE TOTAL LENGTH
BNP GWRETURN NO (SHOULDN'T HAPPEN)
LR R6,R1 POINT TO BEGINNING OF STRING
AR R1,R8 POINT TO DEFAULT END (SHOULDN'T BE)
L R7,=A( TRTTAB2) KEEP FROM BLOWING BASE REGISTER
EX R8,GWTRT2 FIND FIRST BLANK/SPECIAL
LR R7,R1 SET CURRENT POSITION
SR R7,R6 COMPUTE LENGTH OF STRING
BCTR R7,0 LENGTH - 1
GWRETURN L RBAL,SAVGWBAL RESTORE LINKAGE REGISTER
BR RBAL RETURN
GWNULL XR R8,R8 FORCE NULL LENGTH
B GWRETURN EXIT
GWTRT1 TRT Ø(*-*,R6),TRTTAB1
GWTRT2 TRT Ø(*-*,R6),Ø(R7)
EJECT

******************************************************************************
*** THIS IS AN INTERNAL SUBROUTINE TO SCAN CHARACTER STRINGS FOR 'WORDS' (IE ALPHAMERIC SUBSTRINGS). RETURNED FIELDS ARE NON-BLANK CHARACTER STRINGS THAT ARE CONCATENATED BY AT LEAST ONE BLANK OR NON-ALPHAMERIC CHARACTER.***

*———————————————————————————————————————————————————————————————————*
*** TO REDUCE INSTRUCTION PATH LENGTH IT NEITHER SAVES REGISTERS NOR USES CONVENTIONAL CALLING SEQUENCE. ***
*———————————————————————————————————————————————————————————————————*

*** USAGE: ***
*** 1) TO SCAN FOR FIELD SEPARATED BY ' ', ',', '"', '( ' OR ' )' ***
*** LA R4,NULL LOAD ADDRESS OF EOB RETURN ***
*** BAL R14,KHNSCAN SCAN FOR NEXT INPUT FIELD ***
*** 2) TO VALIDATE NUMERIC FIELDS: ***
*** LA R4,ERROR LOAD ADDRESS OF NON-NUMERIC RETURN ***
*** BAL R14,NUMTEST CHECK FIELD FOR NUMERIC DATA ***

* REGISTER USAGE: *
* 1) FOR KHNSCAN, CONTENTS OF REGISTER 1 IS USED AS A WORK REGISTER AND IS NOT RESTORED. *
* 2) ON ENTRY TO KHNSCAN AND NUMTEST, THE FOLLOWING ASSUMPTIONS *
* ARE MADE: REGISTER 6 CONTAINS THE ADDRESS OF THE CURRENT FIELD; REGISTER 7, THE LENGTH - 1 OF THAT FIELD; REGISTER 8, THE REMAINING LENGTH OF THE TIOA.

* 3) ON RETURN, KHNSCAN AND KHNSCAN, REGISTERS 6-8 ARE SET TO THOSE VALUES DEFINED IN “2)”.

* 4) FOR NUMERIC FIELDS, NUMTEST PACKS THE FIELD INTO 'PACKWORK'. ELSE, THIS FIELD IS INITIALIZED TO ZERO.

***********************************************************************

KHNSCAN MVC TRTAb,TRTAB-1  SET TABLE TO NON ZERO
MVI TRTAB+C ',Ø  CLEAR BLANK POSITION
XR R1,R1  CLEAR REGISTER (HIGH ORDER BYTE)
LA R6,1(R6,R7)  POINT PAST LAST FIELD

PRESCAN CLI Ø(R6),C '='  EQUAL SIGN?
BE SPECIAL  YES
CLI Ø(R6),C '+'  PLUS SIGN?
BNE NOTPLUS  NO
MVI SIGN,X'C'  SET SIGN
B SPECIAL  GO ADJUST POSITION AND LENGTH

NOTPLUS CLI Ø(R6),C '-'  MINUS SIGN?
BNE NOTMINUS  NO
MVI SIGN,X'D'  SET SIGN
B SPECIAL  GO ADJUST POSITION AND LENGTH

NOTMINUS CLI Ø(R6),C '('  OPEN PARENTHESSES?
BNE NOTLEFT  NO
AP NESTS,=P'1'  INCREMENT NESTING COUNT
B SPECIAL  GO ADJUST POSITION AND LENGTH

NOTLEFT CLI Ø(R6),C ')'  RIGHT PARENTHESIS?
BNE NOTRIGHT  NO
SP NESTS,=P'1'  DECREMENT NESTING COUNT
B SPECIAL  GO ADJUST POSITION AND LENGTH

NOTRIGHT CLI Ø(R6),C ''''  WAS FIELD FOLLOWED BY A QUOTE?
BNE NOTQUOTE  NO
XI SWITCHES,QUOTEBIT  FLIP QUOTE BIT
B SPECIAL  GO ADJUST POSITION AND LENGTH

NOTQUOTE CLI Ø(R6),C ','  IS CURRENT POSITION A COMMA?
BNE NONSPCL  NO

SPECIAL LA R6,1(R6)  SKIP PAST SPECIAL CHARACTER
BIT R8,Ø  DECREMENT LENGTH
LTR R8,R8  END OF CARD?
BMR R4  YES
B PRESCAN  GO PROCESS NEXT CHARACTER

NONSPCL EX R8,TRT  SEARCH FOR FIRST NON BLANK
BZR R4  EXIT IF NOT FOUND
LR R7,R1  ADDRESS OF FIRST NON-BLANK
SR R7,R6  DEDUCT ADDRESS OF LAST POSITION
SR R8,R7  SUBTRACT LENGTH FROM TOTAL LENGTH
BMR R4  EXIT IF NEGATIVE
LR R6,R1  POINT TO FIRST NON BLANK
CLI Ø(R6),C '''  QUOTATION AT BEGINNING?
BE PRESCAN  YES, RECIRCULATE
CLI Ø(6),C'(') OPEN PAREN AT BEGINNING?
BE PRESCAN YES, RECIRCULATE
CLI Ø(6),C',.' NULL FIELD AT BEGINNING?
BE PRESCAN YES, RECIRCULATE
CLI Ø(6),C'+.' UNARY PLUS SIGN AT BEGINNING?
BE PRESCAN YES, RECIRCULATE
CLI Ø(6),C'-.' UNARY MINUS SIGN AT BEGINNING?
BE PRESCAN YES, RECIRCULATE
XC TRTAB,TRTAB SET TABLE TO ZEROES
MVI TRTAB+C' ','C' ' TURN ON BLANK POSITION
MVI TRTAB+C',',C',.' TURN ON COMMA POSITION
MVI TRTAB+C'''',C'''' TURN ON C'''' POSITION
MVI TRTAB+C'(',C'(' TURN ON C'(' POSITION
MVI TRTAB+C')',C')' TURN ON C')' POSITION
MVI TRTAB+C=',C'=' TURN ON C '=' POSITION
MVI TRTAB+C'+',C'+.' TURN ON C'+.' POSITION
MVI TRTAB+C'-',C'-.' TURN ON C'-.' POSITION
LR R15,R8 SAVE CURRENT LENGTH
LR R0,R6 SAVE CURRENT LOCATION
LASTSCAN EX R8,TRT SEARCH FOR FIRST BLANK
BZ NOHITS IF NO OBJECTS FOUND
TM SWITCHES,QUOTEBIT WITHIN QUOTATION?
BZ SCANHIT NO
CLC =C''''''',Ø(1) IMBEDDED QUOTES?
BNE SCANHIT NO
LA R1,2(R1) STEP OVER IMBEDDED QUOTES
AR R8,R6 ADJUSTED LENGTH=PREVIOUS LENGTH
SR R8,R1 +(PREVIOUS-CURRENT)LOCATION
LR R6,R1 RESET CURRENT POSITION
BP LASTSCAN IF POSITIVE LENGTH, CONTINUE SCAN
LR R6,R0 RESTORE ORIGINAL LOCATION
LR R8,R15 RESTORE ORIGINAL LENGTH
B SCANHIT GO PROCESS
NOHITS LR R6,R0 RESTORE ORIGINAL LOCATION
LR R8,R15 RESTORE ORIGINAL LENGTH
LA R1,Ø(R6,R8) POINT TO END OF INPUT
SCANHIT LR R7,R1 LOAD ADDRESS OF BLANK
SR R7,R6 SUBTRACT ADDRESS OF FIRST NON-BLANK
BCR 13,R4 NULL IF NOT POSITIVE
SR R8,R7 DEDUCT FROM TOTAL LENGTH
BCTR R7,R14 RETURN
BR R14 RETURN
TRT TRT Ø(*-* ,R6),TRTAB
TESTNUM TRT Ø(*-* ,R6),TRTAB+16
NUMTEST MVC TRTAB,TRTAB-1 FILL WITH NON ZEROES
EX R7,TESTNUM IS FIELD NUMERIC?
BCR 7,R4 NO
EX R7,PACK PACK FIELD
NI PACKWORK+L'PACKWORK-1,X'F0' MASK SIGN BITS OFF
OC PACKWORK+L'PACKWORK-1(1),SIGN TURN SIGN BITS ON
BR  R14  RETURN
PACK  PACK  PACKWORK,Ø(*-*,6)
EJECT

******************************************************************************
***  THIS ROUTINE PRINTS DIAGNOSTIC DATA IF 'DIAG' OPTION IS ***
***  SPECIFIED.  IT IS USED PRIMARILY IN TESTING CHANGES TO THE ***
***  PROGRAM OR IN DIAGNOSING ANY PROBLEMS WITH SPECIFIC DATA ***
******************************************************************************

TEST  TM  OPTIONS,DIAGBIT  DIAGNOSE BIT ON?
BZR  RBAL  NO
TESTX  ST  RBAL,SAVTSBAL  SAVE LINKAGE REGISTER
UNPK  TESTOPTS(3),OPTIONS(2) UNPACK NYBLS OF BIT SWITCHES
UNPK  TESTSWTS(3),SWITCHES(2) UNPACK NYBLS OF BIT SWITCHES
ST  R7,DOUBLE  SAVE LENGTH
UNPK  TESTLEN(5),DOUBLE(5) UNPACK NYBLS OF LENGTH
ST  R6,DOUBLE  SAVE LENGTH
UNPK  TESTLOC(5),DOUBLE(5) UNPACK NYBLS OF ADDRESS
NC  TESTOPTS(15),=15'F' TURN OFF ZONE BITS
TR  TESTOPTS(15),=C'Ø123456789ABCDEF' CONVERT TO HEX DSpLY
MVI  TESTSWTS+2,C' ' SET SEPARATOR
MVI  TESTOPTS+2,C' ' SET SEPARATOR
MVI  TESTLEN+4,C' ' "
MVI  TESTLOC+4,C' ' "
MVC  LINE+1(L'LINE-1),INAREA+1 MOVE IMAGE TO PRINT LINE
BAL  RBAL,PRINT  GO PRINT DIAGNOSTIC LINE
TSRETURN  L  RBAL,SAVTSBAL  RESTORE LINKAGE REGISTER
BR  RBAL  RETURN
TSMOVE  MVC  INAREA(*-*),Ø(6)
EJECT

******************************************************************************
***  MOVE DATA LEFT BY AN OFFSET SPECIFIED IN REGISTER R4.  R6 ***
***  POINTS TO CURRENT POSITION.  R2 SPECIFIES THE DESTINATION ***
***  OF THE MOVE. ***
******************************************************************************

MOVELEFT  ST  RBAL,SAVMLBAL  SAVE LINKAGE REGISTER
LA  R14,Ø(R2,R4)  DESTINATION + OFFSET
LR  R15,R6  POINT TO CURRENT POSITION
SR  R15,R2  LENGTH OF MOVE (REPLICATE TARGET)
LTR  R15,R15  IS IT NECESSARY TO MOVE?
BNP  MLNOMOVE  NO
BCTR  R15,Ø  LENGTH - 1 (FOR MVC)
EX  R15,MLMOVE  MOVE LEFT
MLNOMOVE  SR  R6,R4  ADJUST CURRENT POSITION
AR  R7,R4  ADJUST SCAN LENGTH
L  RBAL,SAVMLBAL  RESTORE LINKAGE REGISTER
BR  RBAL  RETURN
MLMOVE  MVC  Ø(*-*,R2),Ø(R14)
EJECT

******************************************************************************
***  MOVE DATA RIGHT ***
** AT ENTRY: R6 = LOCATION OF TARGET STRING  
** R7 = LENGTH-1 OF TARGET STRING  
** R2 = LOCATION OF AVAILABLE SPACE TO COMPRESS  
** R4 = MINIMUM OF BYTES AVAILABLE, NEEDED  
**  
** HENCE: R2-1 = RIGHT MOST BYTE TO MOVE RIGHT (TO R2)  
** R6+R1+1 = LEFT MOST BYTE TO MOVE RIGHT  

**-----------------------------------------------------------------------------**

MOVERGHT ST R2,SAVMRBAL  
LR R15,R2  
LA R14,R6,R7  
LR R1,R15  
SR R1,R4  
SR R1,R14  
BNP MRRETURN  
LA R1,R1(R1)  
LR R14,R15  
SR R1,R4  
MRLOOP MVC Ø(1,R15),Ø(R14)  
BCTR R14,Ø  
BCTR R15,Ø  
BCT R1,MRLOOP  
LA R14(R6,R7)  
AR R7,R4  
MRRETURN L R2,SAVMRBAL  
BR R2  
LTORG  

OCCURS DC C'CONTAINS'  
OCCUR1 DC X'40204B20204B202120'  
DC C' RECORDS OF WHICH'  
OCCUR2 DC X'40204B20204B202120'  
DC C' CONTAIN'  
OCCUR3 DC X'40204B20204B202120',C' '  
DC C'SPECIFIED OCCURRENCES'  
LOCCURS EQU -*OCCURS  
OCCURPAT DC X'402020202120'  
EDITPAT EQU OCCURPAT  
&W3ET A 4  
&P3REFIX SETA 2  
&S3UFFIX SETA 1  
WORDBIT EQU &WORD  
&PREFBIT EQU &PREFIX  
&SUFFIX EQU &SUFFIX  
WORDLIST DS ØC  
PUSH PRINT  
PRINT GEN  
STDEF SPACE,'C'' ''',W  
STDEF ZERO,LOW-VALUE  
STDEF XYZ-DATE,XYZ-NEW-DATE,OPTION=FORCE
STDEF XYZ-YY,XYZ-CCYY
STDEF 'QUOTE''TEST',NEW''QUOTE''TEST
LASTWORD DC X'FF'  
NOTE THAT THIS MUST IMMEDIATELY FOLLOW LIST OF CHARACTER STRINGS
POP PRINT
IMDEF DC AL1(&IMBED)
OTDEF DC AL1(&OTHER)
TRTTAB1 DC 256X'Ø'
ORG TRTTAB1+X'81'  LOWER CASE 'A'
DC X'818283848486878889'
ORG TRTTAB1+X'91'  LOWER CASE 'J'
DC X'919293949596979899'
ORG TRTTAB1+X'A2'  LOWER CASE 'S'
DC X'A2A3A4A5A6A7A8A9'
ORG TRTTAB1+C'@'
DC C'@'
ORG TRTTAB1+C'#'
DC C'#'
ORG TRTTAB1+C'$'
DC C'$'
ORG TRTTAB1+C'A'
DC C'ABCDEFGHI'
ORG TRTTAB1+C'J'
DC C'JKLMNOPQR'
ORG TRTTAB1+C'S'
DC C'STUVWXYZ'
ORG TRTTAB1+C'Ø'
DC C'0123456789'
ORG
TRTTAB2 DC 256X'FF'
ORG TRTTAB2+X'81'  LOWER CASE 'A'
DC 9X'Ø'
ORG TRTTAB2+X'91'  LOWER CASE 'J'
DC 9X'Ø'
ORG TRTTAB2+X'A2'  LOWER CASE 'S'
DC 8X'Ø'
ORG TRTTAB2+C'@'
DC X'Ø'
ORG TRTTAB2+C'#'
DC X'Ø'
ORG TRTTAB2+C'$'
DC X'Ø'
ORG TRTTAB2+C'A'
DC 9X'Ø'
ORG TRTTAB2+C'J'
DC 9X'Ø'
ORG TRTTAB2+C'S'
DC 9X'Ø'
ORG TRTTAB2+C'Ø'
DC 10X'Ø'
ORG
LTORG
EJECT
***********************************************************************
*** SPECIAL INITIALIZING ROUTINE TO CONSERVE BASE REGISTER ***
*** ADDRESSING PAGE ***
***********************************************************************
INITIAL ST RBAI,SAVI BAL SAVE LINKAGE REGISTER
LA R11,2048(RBASE) LOAD RBASE + HALF PAGE
LA R11,2048(R11) LOAD RBASE + FULL PAGE
USING &MYNAME,RBASE,R11 ADDRESSABILITY
MVI OPTIONS,Ø INITIALIZE OPTIONS
ZAP NESTS,=P’Ø’ INITIALIZE ‘(‘,’)’ NESTING COUNT
MVI TRTAB-1,X’FF’ INITIALIZE NON-ZERO PREFIX
MVC TRTAB+L’TRTAB(10),=10X’FF’ SET NON-BLANK FOR NUM TEST
MVC THRUNAME,=19X’FF’ SET INITIAL ‘THRU’ MEMBER NAME
XC FROMNAME,FROMDATE “ ’FROM’ MEMBER NAME
MVI DFLAG,Ø INITIALIZE FLAG
ZAP FINDS,=P’Ø’ INITIALIZE STRING FOUND COUNT
ZAP MEMBERS,=P’Ø’ INITIALIZE MEMBERS IN PDS
ZAP MODIFIED,=P’Ø’ INITIALIZE MODIFIED MEMBERS
ZAP EXCLUDED,=P’Ø’ INITIALIZE EXCLUDED MEMBERS
ZAP RECORDS,=P’Ø’ INITIALIZE RECORDS IN 1ST MEMBER
ZAP STRINGS,=P’Ø’ INITIALIZE STRING OCCURRENCES IN 1ST
ZAP TREC,=P’Ø’ INITIALIZE RECORDS IN ALL MEMBER
ZAP TFIND,=P’Ø’ INITIALIZE TOTAL SELECTED RECS
ZAP TSTRINGS,=P’Ø’ INITIALIZE TOTAL STRING OCCURRENCES
ZAP ERRORTOT,=P’Ø’ INITIALIZE ERRORS IN ALL MEMBERS
MVC JGMOBEL(13*L’JGMOBEL),JGMOBELD COPY JULGREG DAYS/MONTH
* BEGIN DCB INITIALIZATION
MVC PRINTER(PRINTERL),PRINTERD INITIALIZE DCB
MVC PDSDIR(PDSDIRL),PDSDIRD INITIALIZE PDSDIR DCB
MVC CARDS(CARDSL),CARDSD INITIALIZE CARDS DCB
MVC ERRORS(ERRORSL),ERRORSD INITIALIZE ERRORS DCB
* END DCB INITIALIZATION
* BEGIN DCB OPENS
MVC PROPENL(OPENPNL),OPEND INITIALIZE SET PRINTER OPEN LIST
OPEN (PRINTER,(OUTPUT)),MF=(E,PROPENL) OPEN PRINTER
MVC DROOPENL(DROOPENL),OPEND SET PDSDIR OPEN LIST
OPEN (PDSDIR,(INPUT)),MF=(E,DROOPENL) OPEN PDSDIR
MVC PDOPENL(PDOPENL),OPEND SET PDS OPEN LIST
OPEN (PDS,(UPDAT)),MF=(E,PDOPENL) OPEN PDS
MVC EROOPENL(EROOPENL),OPEND SET ERRORS OPEN LIST
OPEN (ERRORS,(OUTPUT)),MF=(E,EROOPENL) OPEN ERRORS
MVC PRCLOSL(PRCLOSL),CLOSED INITIALIZE CLOSE LIST
MVC DRCLOSL(DRCLSL),CLOSED SET PDSDIR CLOSE LIST
MVC PDCLSL(PDCLSL),CLOSED SET PDS CLOSE LIST
MVC ERCLOSL(ERCLOSL),CLOSED SET ERRORS CLOSE LIST
LA R3,PDS GET ADDRESS OF PDS DCB
USING IHADCB,R3           ESTABLISH ADDRESSABILITY
LH   R5,DCBLRECL       LOAD RECORD LENGTH
STH  R5,INLRECL       SAVE
LH   R3,DCBBLKSI       LOAD MAXIMUM BLOCK SIZE
STH  R3,INBLKSIZ      SAVE
LA    R3,1IØØ(R3)      ADD PAD
DROP  R3              DROP ADDRESSABILITY
GETMAIN R,LV=(R3)     GET WORK AREA FOR INPUT BLOCKS
ST    R1,BLOCKLOC     SAVE ADDRESS
MVC   CDOPENL(CDOPENLN),OPEND   SET CARDS OPEN LIST
OPEN  (CARDS,(INPUT)),MF=(E,CDOPENL)   OPEN CARDS
* END DCB OPENS
ZAP   IMBEDDED,=P'Ø'   INITIALIZE 1ST MEMBER IMBEDDED COUNT
MVC   WORDS(1I*L'TOTALS),IMBEDDED " WORD,PREFIX,SUFFIX,1ST STR
LA    R15,TOTALS       POINT TO FIRST TOTAL
LA    RØ,8*L'TOTALS    SIZE OF ENTRY
LA    R1,GRANDS        POINT TO GRAND TOTALS
ILTOTALS MVC    8*L'TOTALS(8*L'TOTALS,R15),Ø(R15) " NEXT LINE
BXLE  R15,RØ,ILTOTALS  CONTINUE
TIME
ST    R1,JGYYDDD       SAVE JULIAN DATE
ST    R1,TODAY         SAVE FORM PARM DATA
BAL   RBAL,JULGREG     CONVERT TO MM/YY/DD
MVC   HEADER(L'HEAD),HEAD INITIALIZE HEADER
MVC   HEADER+L'HEADER-L'HEAD),HEADER+L'HEAD-1 CLEAR
MVC   PAGENO-4(4),=C'PAGE' SET PAGE NUMBER ID
ZAP   PAGES,=P'1'      INITIALIZE PAGE COUNT
MVC   DDNAME,PDSDDN    MOVE SELECTION FILE NAMES
BAL   RBAL,GETNAMES    PUT JOB/DSN NAMES IN HEADER
MVC   HEADDATE,JGMDCY   MOVE MM/DD/CCYY TO HEADING
BAL   RBAL,HEADPAGE    PRINT PAGE HEADER
BAL   RBAL,GETPARMS    GET PARMS
MVC   DECB(DECBALS),DECBD INITIALIZE DECB
LA    R3,PDS           GET ADDRESS OF PDS DCB
USING IHADCB,R3        ESTABLISH ADDRESSABILITY
LH   R5,DCBLRECL       LOAD RECORD LENGTH
STH  R5,INLRECL       SAVE
LH   R3,DCBBLKSI       LOAD MAXIMUM BLOCK SIZE
STH  R3,INBLKSIZ      SAVE
LA    R3,1IØØ(R3)      ADD PAD
DROP  R3              DROP ADDRESSABILITY
GETMAIN R,LV=(R3)     GET WORK AREA FOR INPUT BLOCKS
ST    R1,BLOCKLOC     SAVE ADDRESS
LA    R3,EXCLUDES      POINT TO FIRST ELEMENT
LA    R4,EXCLUDEX-EXCLUDES(R3) POINT TO LAST EXCLUDE
ST    R3,EXCLUDE1     SAVE BEGINNING ADDRESS
MVC   LINE(27),=C'ØMANUALLY EXCLUDED MEMBERS:'
BAL   RBAL,DOUBLESP   ALLOW FOR DOUBLE SPACE
BAL   RBAL,PRINT      PRINT EXCLUSION SUBHEADER
MVI   LINE,C'Ø'       SET TO DOUBLE SPACE
BAL    RBAL,DOUBLESP  ALLOW FOR DOUBLE SPACE

ILCDLOOP GET       CARDS,CARDAREA  READ EXCLUSION CARD
MVC Ø(L'EXCLUDES,R3),CARDAREA MOVE MEMBER NAME TO EXCL TABLE
LA    R3,L'EXCLUDES(R3)  POINT TO NEXT ENTRY
CR    R3,R4             PAST END OF SAVE AREA?
BL    ILCDLOOP           NO

CARDEOF MVC CDCLOSL(CDCLOSLN),CLOSED SET CARDS CLOSE LIST
CLOSE (CARDS),MF=(E,CDCLOSL)  CLOSE CARDS
MVC Ø(L'EXCLUDES,R3),=8X'FF' SET HIGH VALUES
ST    R3,EXCLUDE2       SAVE LAST CARD IMAGE
C     R3,EXCLUDE1       ANY EXCLUSIONS?
BNE   ILSORT            NO
MVC   LINE+5(8),=C'* NONE *' INDICATE NO EXCLUSIONS
BAL   RBAL,PRINT       PRINT INDICATION
B     ILEXIT            GO EXIT

ILSORT L  R3,EXCLUDE1 LOAD START OF LIST
ILSORTL2 LA    R4,L'EXCLUDES(R3) PART TO NEXT ELEMENT OF VECTOR
C     R4,EXCLUDE2       AT END OF VECTOR?
BE    ILSORTX2         YES (BUT PRINT LAST ENTRY)
BH    ILEXIT            YES

ILSORTL1 CLC  Ø(L'EXCLUDES,R4),Ø(R3) CURRENT ENTRY LOWER?
BH    ILSORTX1         NO
XC  Ø(L'EXCLUDES,R3),Ø(R4) SWAP
XC  Ø(L'EXCLUDES,R4),Ø(R3) . VECOR
XC Ø(L'EXCLUDES,R3),Ø(R4) . ELEMENTS

ILSORTX1 LA    R4,L'EXCLUDES(R4) PART TO NEXT ENTRY
C     R4,EXCLUDE2       AT END OF LIST?
BL    ILSORTL1         NO

ILSORTX2 MVC   LINE+5(L'EXCLUDES),Ø(R3) MOVED SORTED ENTRY
BAL   RBAL,PRINT       PRINT ENTRY
LA    R3,L'EXCLUDES(R3) PART TO NEXT ENTRY
B     ILSORTL2         CONTINUE
ILEXIT   MVI   LINE,C'Ø' SET TO DOUBLE SPACE
BAL    RBAL,DOUBLESP   ALLOW FOR DOUBLE SPACE
L     RBAL,SAVILBAL    RESTORE LINKAGE REGISTER
BR    RBAL             RETURN
EJECT

**********************************************************************
***   GET JOB AND PDS DSN NAMES                                     ***
*** ——————————————————————————————————————————————————————————— ***
***   THANKS TO MR MARK HOFFMAN FOR THIS LOGIC                      ***
**********************************************************************

GETNAMES ST    RBAL,SAVGNBAL SAVE LINKAGE REGISTER
XR    R15,R15          ADDRESS OF PSA
USING  PSA,R15         ESTABLISH ADDRESSABILITY
L     R14,FLCCVT       ADDRESS OF CVT
DROP   R15             DROP ADDRESSABILITY TO PSA
USING  CVTMAP,R14      ESTABLISH ADDRESSABILITY TO CVT
L     R15,CVTTCBP      ADDRESS OF NEXT TCB POINTER
L     R15,4Ø,R15      ADDRESS OF CURRENT TCB

© 1998. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (940) 455 7050, fax (940) 455 2492.
DROP R14
USING TCB,R15
L R14,TCB
USING TCIOT,R14
MVC HEADJOBN,TIOCNJOB
MVC HEADJOBN-4(4),=C'JOB=' SET JOBNAME ID
DROP R15
LA R15,TIOELNGH ADDRESS OF FIRST TIOT ENTRY
DROP R14
USING TIOENTRY,R15
GNTIOTLP CLI TIOELNGH,X'000' END OF TIOT CHAIN?
BE GNRETURN YES (SHOULDN'T HAPPEN)
CLC TIOEDDNM(8),DDNAME PDS NAME FOUND?
BE GNDSN YES
XR R8,R8 CLEAR REGISTER
IC R8,TIOELNGH INSERT ENTRY LENGTH
AR R15,R8 POINT TO NEXT ENTRY
B GNTIOTLP CONTINUE
GNDSN XR R1,R1 CLEAR REGISTER
ICM R1,7,TIOEJFCB ADDRESS OF JFCB
USING JFCB,R1 ESTABLISH ADDRESSABILITY TO JFCB
MVC HEADDSN,JFCBDSNM MOVE DSNAME TO HEADER
MVC HEADDSN-4(4),=C'DSN=' SET DSN ID IN HEADER
DROP R1,R15 DROP ADDRESSING TO JFCB,TIOT,ENTRY
GNRETURN L RBAL,SAVGNBAL RESTORE LINKAGE REGISTER
BR RBAL RETURN

***********************************************************************
***   ANALYZE 'PARM=' DATA                                          ***
***********************************************************************
GETPARMS ST RBAL,SAVGPBAL SAVE LINKAGE REGISTER
L R6,R1SAVE GET ADDRESS OF AREA
L R6,R6(R6) GET ADDRESS OF PARM= DATA
LH R8,R8(R6) LOAD LENGTH OF PARM
LTR R8,R8 VALID LENGTH?
BNP GPRETURN NO
LA R6,1(R6) POINT TO BYTE PRECEDING INFO FIELD
XR R7,R7 CLEAR INITIAL LENGTH
MVC LINE+1(5),=C'PARM=' SET IDENTIFIER
LR R1,R8 LENGTH OF PARM= STRING
BCTR R1,0 LENGTH - 1
EX R1,MVEPPARM MOVE PARAMETERS TO PRINT LINE
BAL RBAL,PRINT GO PRINT PARAMETERS
MVI LINE,C'0' SET TO DOUBLE SPACE
BAL RBAL,DUBLESPO ALLOW FOR DOUBLE SPACE
PARMLOOP LA R4,PARMEND POINT TO NULL RETURN
BAL R14,KHNSCAN GET PARAMETER
BAL RBAL,TEST FOR TESTING
NOTSUBPM LA R4,PARMERR POINT TO NULL RETURN
CLC =C'PRNT=','R8 'PRINT' OPTION?
BE   SETPRINT, YES
CLC  =C'FROM=',Ø(R6)  'FROM' OPTION?
BE   SETFROM, YES
CLC  =C'THRU=',Ø(R6)  'THRU' OPTION?
BE   SETTHRU, YES
CLC  =C'CTRL=',Ø(R6)   'CONTROL' OPTION?
BNE  PARMERR, NO
MVI   SIGN,X'F'       INITIALIZE SIGN (NOT REALLY NECESRY)
BAL  R14,KHNSCAN  GO GET CONTROL VALUE
CLI  Ø(R6),C'X'   OVERRIDE?
BE   OVERRIDE, YES
BAL  RBAL,TEST  FOR TESTING
BAL  R14,NUMTEST  VERIFY THAT IT'S NUMERIC
ZAP  CONTROL,PACKWORK  SET VALUE
B    PARMLOOP  CONTINUE PARAMETER SCAN
OVERRIDE MVI  CONTROL,X'FF'   SET OVERRIDE
B    PARMLOOP  CONTINUE PARAMETER SCAN
PRINT4 CLI  4(R6),C',,' ADDITIONAL PRINT OPTION?
BE   SETPRINT, YES
B    PARMLOOP  NO
DIAGNOSE OI OPTIONS,DIAGBIT  TURN ON OPTION
B    PRINT4  GO CHECK FOR ADDITIONAL PRNT OPTS
BEFORE OI OPTIONS,BFOREBIT  TURN ON OPTION
CLI  6(R6),C',,' ADDITIONAL PRINT OPTION?
BE   SETPRINT, YES
B    PARMLOOP  NO
AFTER OI OPTIONS,AFTERBIT  TURN ON OPTION
CLI  5(R6),C',,'  ADDITIONAL PRINT OPTION?
BE   SETPRINT, YES
B    PARMLOOP  NO
SETFROM BAL  R14,KHNSCAN  GET PRINT OPTION
BAL  RBAL,TEST  FOR TESTING
MVC  MEMBER,=8C' '  INITIALIZE NAME PADDING
EX  R7,MOVENAME  MOVE MEMBER NAME
OI OPTIONS,MEMRBiT  SET OPTION BIT
MVC  FROMNAME,MEMBER  MOVE TO BEGINNING NAME
B    PARMLOOP  NO
SETTHRU BAL  R14,KHNSCAN  GET PRINT OPTION
BAL  RBAL,TEST  FOR TESTING
MVC MEMBER.=BC' ' INITIALIZE NAME PADDING
EX R7,MOVENAME MOVE MEMBER NAME
OI OPTIONS,MEMBRBIT SET OPTION BIT
MVC THRUNAME,MEMBER MOVE TO ENDING NAME
B PARMLOOP NO
MOVENAME MVC MEMBER(*-*),Ø(6)
PARMEND CLI CONTROL,X'FF' OVERRODE?
PARMERR DS ØH FOR NOW
GPRETURN L RBAL,SAVPB AL RESTORE LINKAGE REGISTER
BR RBAL RETURN
MOVEPARM MVC LINE+6(*-*),1(R6)
* END STUB DEFINE
EJECT
**********************************************************************
***      FIXED DATA AREA                                           ***
**********************************************************************
HEAD DC C'1YEAR2KRS — REPLACE CHARACTER STRINGS '
* CHECK REFERENCES TO THE FOLLOWING EQUATES IF CHANGES ARE MADE TO
* TABLE.
FMTCYMD EQU 4 FORMATS WITH CCYY AT BEGINNING =>
FMTYMD EQU 2 LOGIC USES THIS IN PARM/DATA ANALYSIS
FORMATS DC CLB'CCYY/MM/DD','X'9'  4
DC CLB'MM/DD/CCYY','X'9'  3
DC CLB'YY/MM/DD','X'7'    2
DC CLB'MM/DD/YY','X'7'    1
#FORMATS EQU (*-FORMATS)/(L'FORMATS+1)
OPEND OPEN (,),MF=L
CLOSED CLOSE (,),MF=L
READ DECBD,SF,MF=L
* BEGIN DCB CONSTANTS
PRINTERD DCB DDNAME=PRINTER,DEVD=DA,DSORG=PS,LRECL=133,
- BLKSIZE=133,MACRF=(PM),RECFM=FBA
PDSDIRD DCB DDNAME=PDS,DSORG=PS,MACRF=GM,BLKSIZE=256,LRECL=256,
- EODAD=GDEND,RECFM=F
PDS DD CB DDNAME=PDS,DSORG=PO,MACRF=R,EODAD=GREOF
PDSDDN EQU PDSDIRD+DCBDDNAM-DCBRELAD
CARDSD DCB DDNAME=CARDS,DSORG=PS,MACRF=GM,EODAD=CARDEOF,
- RECFM=FB,LRECL=80
ERRORSD DCB DDNAME=ERRORS,DEVD=DA,DSORG=PS,LRECL=133,
- BLKSIZE=133,MACRF=(PM),RECFM=FBA
* END DCB CONSTANTS
* END CONSTANTS
LTORG
EJECT
**********************************************************************
***      DSECT FOR MY SAVE AREA AND VARIABLES.                    ***
**********************************************************************
WORKD DSECT
MYSAVE DS 18F MY REGISTER SAVE AREA
COMPCODE DS F                        PROGRAM COMPLETION CODE
RETCDE DS F                        INTERNAL RETURN CODE
RSAVE DS F                        INITIAL VALUE IN R1
BLOCKLOC DS F
BLOCKEND DS F
INLRECL DS H
INBLKSIZ DS H
INRECLOC DS F
TTRN DS F
PAGES DS PL2
HIT DS C
DFLAG DS C
NESTS DS PL2
ERRORTOT DS PL3
MEMBERS DS PL3
MODIFIED DS PL3
EXCLUDED DS PL3
RECORDS DS PL4
STRINGS DS PL4
TRECS DS PL4
TFINDS DS PL4
TSTRINGS DS PL4
MEMBER DS CL8
FROMNAME DS XL8'Ø'
THRUNAME DS XL8'FFFFFFFFFFFFFFFF'
NEWDATE DS CL1Ø
OPTIONS DC X'Ø'
CHNGBIT EQU X'4Ø'
DIAGBIT EQU X'2Ø'
LISTBIT EQU X'1Ø'
BFOREBIT EQU X'Ø8'
AFTERBIT EQU X'Ø4'
ALRDYBIT EQU X'Ø1'
MEMBRBIT EQU X'Ø2'
SWITCHES DC X'Ø'
QUOTEBIT EQU X'8Ø'
COMMABIT EQU X'4Ø'
CONTBIT EQU X'2Ø'
PARMBIT EQU X'Ø8'
UPDATBIT EQU X'Ø4'
DUOBIT EQU X'Ø2'
ERRORBIT EQU X'Ø2'
DATEBIT EQU X'01'
DLENGTH DS X
SAVEFLAG DS X
CONTROL DC PL2'Ø'
LEAPFLAG DC X'Ø'
SIGN DC X'C'
DAYS DS PL2
MONTHS DS PL2
YEARS DS D
SAVEDAYS DS D
MMDDYY DS CL8
DATE DS C' MM/DD/CCYY'
TODAY DS F
AVSP1 DS 2ØCL6
AVSP2 DS A,H
SAVRETOS DS 5F
FROMDATE DS CL8
THRUDATE DS CL8
DDNAME DS CL8
DOUBLE DS D
PACKWORK DS PL16
FSREGSAV DS 7F
* BEGIN STUB LINK SAVE
SAVFSBAL DS A BAL REGISTER SAVE AREA FOR FINDSP
SAVGBAL DS A BAL REGISTER SAVE AREA FOR GETDIR
SAVGNBAL DS A BAL REGISTER SAVE AREA FOR GETNAMES
SAVGPBAL DS A BAL REGISTER SAVE AREA FOR GETPARMS
SAVGRBAL DS A BAL REGISTER SAVE AREA FOR GETREC
SAVGSBAL DS A BAL REGISTER SAVE AREA FOR GETSTATS
SAVGWBAL DS A BAL REGISTER SAVE AREA FOR GETWORD
SAVJGBAL DS A BAL REGISTER SAVE AREA FOR JULGREG
SAVILBAL DS A BAL REGISTER SAVE AREA FOR INITIAL
SAVMLBAL DS A BAL REGISTER SAVE AREA FOR MOVELEFT
SAVMRBAL DS A BAL REGISTER SAVE AREA FOR MOVERIGHT
SAVPBAL DS A BAL REGISTER SAVE AREA FOR PUTERR
SAVPSBAL DS A BAL REGISTER SAVE AREA FOR PUTSTATS
SAVRDBAL DS A BAL REGISTER SAVE AREA FOR READDR
SAVRPBAL DS A BAL REGISTER SAVE AREA FOR REPLACEDIR
SAVS1BAL DS A BAL REGISTER SAVE AREA FOR SCAN1
SAVS2BAL DS A BAL REGISTER SAVE AREA FOR SCAN2
SAVTBSBAL DS A BAL REGISTER SAVE AREA FOR TEST
SAVWRBAl DS A BAL REGISTER SAVE AREA FOR WRITEREC
* END STUB LINK SAVE
* BEGIN OPEN/CLOSE LIST
DS ØD
PROPENL OPEN (,),MF=L
PROPENLN EQU -*-PROPENL
PRCLOS1 CLOSE (,),MF=L
PRCLOS1N EQU -*-PRCLOS1
DROPN1 OPEN (,),MF=L
DROPNLN EQU -*-DROPN1
DRCLOS1 CLOSE (,),MF=L
DRCLOS1N EQU -*-DRCLOS1
PDOPENL OPEN (,),MF=L
PDOPENLN EQU -*-PDOPENL
PDCLOSES1 CLOSE (,),MF=L
PDCLOSES1N EQU -*-PDCLOSES1
CDOPENL OPEN (,),MF=L
CDOPENLN EQU  *-CDOPENL
CDCLOSL CLOSE (),MF=L
CDCLSNLN EQU  *-CDCLOSL
EROPENL OPEN (),MF=L
EROPENLN EQU  *-EROPENL
ERCLOSEL CLOSE (),MF=L
ERCLSNLN EQU  *-ERCLOSEL
* END OPEN/CLOSE LIST
BLDLNTRY SMUMØØ2 DSECT=NO               BLDL FORMAT ENTRY
BLDLLEN EQU   *-BLDLNTRY                 LENGTH OF BLDL ENTRY
READ DECBA,SF,MF=L          DECB FOR PDS
DECBALN EQU   *-DECBA
* BEGIN DCB DSECTS
PRINTER DCB   DDNAME=PRINTER,DEVD=DA,DSORG=PS,LRECL=133,             -
             BLKSIZE=133,MACRF=(PM),RECFM=FBA
PRINTERL EQU   *-PRINTER
PDSDIR DCB   DDNAME=PDS,DSORG=PS,MACRF=GM,BLKSIZE=256,LRECL=256,     -
             EODAD=GDEND,RECFM=F
PDSDIRL EQU   *-PDSDIR
PDS EQU   *-PDS
CARDS DCB   DDNAME=CARDS,DSORG=PS,MACRF=GM,EODAD=CARDEOF,           -
            RECFM=FB,LRECL=8Ø
CARDSL EQU   *-CARDS
ERRORS DCB   DDNAME=ERRORS,DEVD=DA,DSORG=PS,LRECL=133,               -
            BLKSIZE=133,MACRF=(PM),RECFM=FBA
ERRORSL EQU   *-ERRORS
* END DCB DSECTS
JGMOTBL DS   PL2'Ø'
JANUARY DS   P'31'
*                    M  A  M  J  J  A  S  O  N
FEBRUARY DS   P'28,31,30,31,30,31,30,31,30,31,30,31,30,31,30'
DECEMBER DS   P'31'
JGDAYS DS   PL2
JGMONTHS DS   PL2
JGMMDDYY DS   C'MM/DD/YY'
JGMDCY DS   C'MM/DD/CCYY',C
JGYYDDD DS   F
* END DSECT INSERT
HEADER DS   CL133
ORL HEADER+L'HEAD+1Ø
HEADJOBN DS   CL8,C' DSN='
HEADDSN DS   CL44,5C
HEADDRATE DS   CL1Ø
          ORL HEADER+L'HEADER-5
PAGENO DS   CL4
OR
LINE     DS    CL133
OUTAREA  DC    CL133'Ø'
            ORG OUTAREA+2
MEMBERNO   DS    CL4,C
MEMBNAME   DS    CL8
CARDNO     DS    CL6,C
INAREA     DS    CL8Ø,C
TESTOPTS   DS    CL2,C
TESTSWTS   DS    CL2,C
TESTLEN    DS    CL4,C
TESTLOC    DS    CL4,C
            ORG
DIRENTRY DS    F                POINTER TO DIRECTORY ENTRY
DIRSPACE  DS    H                SPACE IN DIRECTORY BLOCK
DIRBLOCK  DS    CL256
FINDS    DS    CL4
IMBEDDED DS    PL3
WORDS    DS    PL(L'IMBEDDED)
PREFIXS  DS    PL(L'IMBEDDED)
SUFFIXS  DS    PL(L'IMBEDDED)
TOTALS   DS    ØPL(L'IMBEDDED)
            .TOTALS ANOP
&N       SETA  &N-1
            DS  ØPL(L'TOTALS)
            AIF (&N GT Ø).TOTALS
GRANDS   DS    ØPL(L'TOTALS)
EXCLUDE1 DS    F
EXCLUDE2 DS    F
CARDAREA DS    CL8Ø
            DS    C'FF'
TRTAB    DS    CL256
            MUST IMMEDIATELY FOLLOW NON-ZERO
            DS    1ØX'Ø'
EXCLUDES DS    3ØØCL8
EXCLUDEX DS    CL8
            DS    ØD
WORKDLEN EQU   *-WORKD
            PRINT GEN
IHAPSA     MAP OF PSA  DSECT=PSA
IKJTTCB    MAP OF TCB  DSECT=TCB
TIOT       DSECT
IEFTTIOT1  MAP OF TIOT
CVT        DSECT=YES    MAP OF CVT  DSECT=CVTMAP
JFCB       DSECT
            MAP OF JFCB
JFCBPREF   DS    CL16
            PREFIX
IEFJFCBN LIST=NO    JFCB PROPER
DCBD     DSORG=PO,DEVD=DA
            A.T.
EJECT
***********************************************************************
***      REGISTER EQUATES                                           ***
***********************************************************************

Checking job datasets exist before job submission

INTRODUCTION
Before submitting a job, I wanted to build a step which contained a reference to all the datasets in the job (in case any were archived or did not exist). SC7 is an edit macro which builds a job step containing DD cards for all the datasets referenced in a job. It will also indicate whether the datasets exist. There are certain restrictions to using the macro, and these are detailed below. The macro cannot handle the following conditions:

• JCL cards that have embedded comments at the end of the line:
  
  //DD DISP=SHR,DSN=MY.DATASET          THIS IS THE INPUT DATASET
  
  The dataset name will be processed as ‘MY.DATASET THIS IS THE INPUT DATASET’, which is clearly invalid!

• JCL cards which point to a dataset name on a tape. To prevent this from happening, you can exclude these lines by adding the word TAPE (or similar for your site) to the exclude list.

There is an ability in the macro to exclude lines from processing that contain certain keywords. These keywords are specified in the ign array – just keep adding to the list! Make sure you update the value
of ign.0 as well. Edit the macro and look for EXCLUDE LIST.

If you want to write out some key values as the macro is processing, type on the command line SC7 < REP2. If you want to trace the macro use SC7 < TEST.

The job step will be built where ever the cursor is positioned when the command is executed (ie type SC7 on the command line, position the cursor to the appropriate place in your job and press enter). If the cursor is on the command line when the enter key is pressed, then the step will be built after 4 lines (this is determined by variable xx in the macro).

```rexx
/* REXX*/
'ISREDIT MACRO (BIGLIN)'
upper biglin

/*************************************************************************/
/* To go through a job and find (and resolve) all DSN= references */
/* These are then put all together in a single step. The location */
/* of this step in the job is determined by the cursor position. */
/*************************************************************************/
If(pos(biglin,'HELP') ¬= Ø 3 biglin = '?') then Do
  say "This edit macro will go thru a job and try and resolve all  "
  say "the DSN= references and put them all in a single step.  "
  say "  "
  say "To invoke this macro position the cursor where you want the"
  say "step containing all the dataset references to be inserted.   "
  say "DO NOT leave the cursor on the command line.            "
ms1 = "Press PF1"
ms2 = ,
"SC7 < REP2"
ZEDSMMSG = ms1
ZEDLMSG = ms2
"ISPEXEC SETMSG MSG(ISRZ000)"
exit
End /* If(pos(biglin,'HELP') ¬= Ø 3 biglin = '?') then Do */
parse var biglin .'<' testflag .
itest = Ø
If(testflag = 'TEST') then Do
  itest = 1
End /* If(testflag = 'TEST') then Do */
If(testflag = 'REP2') then Do
  itest = 2
End /* If(testflag = REP2) then Do */
trace n
If(itest = 1) then Do
  trace r
```
End /* If(itest = 1) then Do */
'ISREDIT (LRECL) = DATA_WIDTH'
'ISREDIT (FIRP) = LINENUM .ZF'
'ISREDIT (LASP) = LINENUM .ZL'
'ISREDIT (ROW,COL) = CURSOR'
parn. = Ø
rowi = row
coli = col
If(itest = 1) then Do
   say 'row   is ' row
   say 'col   is ' col
   say 'zf    is ' firp
   say 'zl    is ' lasp
   say 'lrecl is ' lrecl
End /* If(itest = 1) then Do */
If(row = 1) then Do
   row = 4
End /* If(row = 1) then Do */
iflag3 = Ø
Select
   When(row = 1 & col = Ø ) then Do
      zedsmsg = 'Top Exceeded'
      zedlmsg = 'The cursor was positioned above the first data line.'
      'ISPEXEC SETMSG MSG(ISRZØØ1)'
      iflag3 = 1
   End /* When(row = 1 & col = Ø ) then Do */
   When(row = lasp & col > lrecl ) then Do
      zedsmsg = 'Bot Exceeded'
      zedlmsg = 'The cursor was positioned below the last data line.'
      'ISPEXEC SETMSG MSG(ISRZØØ1)'
      iflag3 = 1
   End /* When(row = lasp & col > lrecl ) then Do */
   Otherwise Nop
End /* Select */
If(iflag3 = 1) then Do
   row = 1
   col = 1
   'ISREDIT CURSOR = (ROW,COL)'
   exit
End /* If(iflag3 = 1) then Do */
/* Read in the whole job to find the DSN names */
'ISREDIT (LRECL) = DATA_WIDTH'
'ISREDIT (FIRP) = LINENUM .ZF'
'ISREDIT (LASP) = LINENUM .ZL'
'ISREDIT (ROW,COL) = CURSOR'
'ISREDIT NULLS = ON'
'ISREDIT (ROW,COL) = CURSOR'
'ISREDIT (LASP) = LINENUM .ZL'
If(itest = 2) then Do
   say TIME() 'Started reading in file'
End /* If(itest = 2) then Do */
row = 1
/***********************************************************/
/* No tracing of these rows.  */
/***********************************************************/
trace n
Do jk = 1 to lasp
   'ISREDIT CURSOR = (ROW,COL)'
   'ISREDIT (STM) = LINE.ZCSR'
   linj.jk = stm
   row = row + 1
End /* Do jk = 1 to lasp */
If(itest = 1) then Do
   trace r
End /* If(itest = 1) then Do */
linj.Ø = lasp
If(itest = 2) then Do
   say TIME() 'Finish  reading in file'
End /* If(itest = 2) then Do */
/***********************************************************/
/* Strip out comment lines. */
/***********************************************************/
If(itest = 2) then Do
   say TIME() 'Started stripping out comment lines'
End /* If(itest = 2) then Do */
jk = Ø
Do jj = 1 to linj.Ø
   If(substr(linj.jj,1,3) = "//") then Do
      Nop
   End /* If(substr(linj.jk,1,3) = */
   Else Do
      jk = jk + 1
      linp.jk = linj.jj
   End /* If(substr(linj.jk,1,3) = hen Do */
End /* Do jj = 1 to linj.Ø */
If(itest = 2) then Do
   say TIME() 'Finish stripping out comment lines'
End /* If(itest = 2) then Do */
If(itest = 2) then Do
   say TIME() 'Started getting proc information'
End /* If(itest = 2) then Do */
/***********************************************************/
/* the main loop */
/***********************************************************/
linp.Ø = jk
jv = Ø
jk = Ø
ip = Ø /* The number of proc sections. */
ie = Ø /* The number of exec statements */
dn. = Ø /* The number of DSNs in each PROC section */
jd = Ø
Do while linp.Ø >= jk
Select
   When (substr(linp.jk,1,2) = '//' & ,
      subword(linp.jk,2,1) = 'PROC') then Do
      ip = ip + 1
      jd = Ø /* The number of datasets within each procedure */
      parse var linp.jk '//' procnam.ip 'PROC'
      If(itest = 2) then Do
         say 'proc ' ip ' is' procnam.ip
         say 'jk is ' jk
      End /* If(itest = 2) then Do */
      ic = Ø
      rest = subword(linp.jk,3)
      If(lastpos(',',rest) > Ø) then Do
         ic = 1
      End /* If(lastpos(',',rest) > Ø) then Do */
      Else Do
         ic = Ø
      End /* If(lastpos(',',rest) > Ø) then Do */
      iflag1 = Ø
      iflag2 = Ø
      If(ic = 1) then Do
         iflag1 = 1
         iflag2 = 1
      End /* If(ic = 1) then Do */
      jp = Ø
      If(iflag1 = Ø) then Do
         jk = jk + 1
      End /* If(iflag1 = Ø) then Do */
      Do while iflag1 = 1
         Do until rest = '
            jp = jp + 1
            parse var rest par.ip.jp '=' pal.ip.jp ',' rest
         End /* Do until rest = ' */
         parn.ip = jp
      If(iflag2 = 1) then Do
         iflag1 = 1
      End /* If(iflag2 = 1) then Do*/
      Else Do
         iflag1 = 0
         End /* If(iflag2 = 1) then Do*/
      If(ic = 1) then Do
         jk = jk + 1
      rest = subword(linp.jk,2)
      If(lastpos(',',rest) = Ø) then Do
         ic = Ø
      iflag2 = Ø
   End /* When (substr(linp.jk,1,2) = '//' & ,
      subword(linp.jk,2,1) = 'PROC') then Do */
End /* If(lastpos(',',rest) = Ø) then Do */
   ic = 1
End /* If(lastpos(',',rest) = Ø) then Do */
End /* If(ic = 1) then Do */
Else Do
   jk = jk + 1
End /* If(ic = 1) then Do */
End /* Do while iflag1 = 1 */
End /* If(substr(linp.jk,1,2) = '//' & , */
When(subword(linp.jk,2,1) = 'DD') then Do
   ipass2 = 1
   If(pos('DSN=',linp.jk) = Ø) then Do
      ipass2 = Ø
   End /* If(pos('DSN=',linp.jk) = Ø) then Do */
   parse var linp.jk 'DSN=' rubb
   If(substr(rubb,1,2) = '&&') then Do
      ipass2 = Ø
   End /* If(substr(rubb,1,2) = '&&') then Do */
   If(ipass2 = Ø) then Do
      Nop
   End /* If(pos('DSN=',linp.jk) = Ø) then Do */
   Else Do
      If(ip = Ø) then Do
         ip = 1
         iflag5 = 1
         parn.1 = Ø
      End /* If(ip = Ø) then Do */
      jd = jd + 1
      parse var linp.jk 'DSN=' rest
      parse var rest ds.ip.jd ',' rubb
      ds.ip.jd = strip(ds.ip.jd,B,"'"")
      ds.ip.jd = strip(ds.ip.jd,B,""")
      If(itest = 2) then Do
         say "DSN reference found: " ds.ip.jd
      End /* If(itest = 2) then Do */
      dn.ip = jd
   End /* If(pos('DSN=',linp.jk) = Ø) then Do */
   jk = jk + 1
End /* If(subword(linp.js,2,1) = 'DD') then Do */

/* Look for the exec card */
/* The format is: //DUM1 EXEC EDA1,PREFIX1='USERID' */

When(substr(linp.jk,1,2) = '//' & ,
   subword(linp.jk,2,1) = 'EXEC',
   & substr(subword(linp.jk,3,1),1,4) = 'PGM=') then Do
   ie = ie + 1
   If(pos('','',linp.jk) = Ø) then Do
      parse var linp.jk 'EXEC' execnam.ie
      rest = ' ' /* No overrides */
```plaintext
End /* If(pos('.', linp.jk) = Ø) then Do */
Else Do
    parse var linp.jk 'EXEC' execnam.ie '',' rest /* Some overs*/
End /* If(pos('.', linp.jk) = Ø) then Do */
If(itest = 2) then Do
    say 'EXEC name ' ie 'is ' execnam.ie
End /* If(itest = 2) then Do */
ic = Ø
rest = strip(rest.T,' ')
If(lastpos('','rest) = length(rest)) then Do
    ic = 1 /* There are parameters on the following line */
End /* If(lastpos('','rest) > Ø) then Do */
Else Do
    ic = Ø /* No parameters on the following line, point to it*/
jk = jk + 1
End /* If(lastpos('','rest) > Ø) then Do */
je = Ø
Do until rest = '
    je = je + 1
    parse var rest ear.ie.je '=' eal.ie.je '',' rest
    If(itest = 2) then Do
        say "ear" ie je "is" ear.ie.je "eal" ie je "is" eal.ie.je
    End /* If(itest = 2) then Do */
End /* Do until rest = '' */
earn.ie = je
Do while ic = 1
    jk = jk + 1
    rest = subword(linp.jk,2)
    rest = strip(rest.T,' ')
    If(lastpos('','rest) = length(rest)) then Do
        ic = 1
    End /* If(lastpos('','rest) > Ø) then Do */
Else Do
    ic = Ø
    jk = jk + 1
End /* If(lastpos('','rest) > Ø) then Do */
Do until rest = '
    je = je + 1
    parse var rest ear.ie.je '=' eal.ie.je '',' rest
    If(itest = 2) then Do
        say "ear" ie je "is" ear.ie.je "eal" ie je "is" eal.ie.je
    End /* If(itest = 2) then Do */
End /* Do until rest = '' */
earn.ie = je
End /* Do while iflag1 = 1 */
End /* If(substr(linp.jk,1,2) = '//' & , */
Otherwise Do
    jk = jk + 1
End /* Otherwise Do */
End /* Select */
```

End /* Do while linp.Ø > jk */

Do x = 1 to ip
  Do jk = 1 to parn.x
    par.x.jk = strip(par.x.jk,B,'"')
    par.x.jk = strip(par.x.jk,B,'"')
    pal.x.jk = strip(pal.x.jk,B,'"')
    pal.x.jk = strip(pal.x.jk,B,'"')
    If(itest = 2) then Do
      say "par: " par.x.jk "value is:" pal.x.jk
    End /* If(itest = 2) then Do */
  End /* Do jk = 1 to jv */
End /* Do jk = 1 to jv */

Do x = 1 to ie
  Do jk = 1 to earn.x
    ear.x.jk = strip(ear.x.jk,B,'"')
    ear.x.jk = strip(ear.x.jk,B,'"')
    eal.x.jk = strip(eal.x.jk,B,'"')
    eal.x.jk = strip(eal.x.jk,B,'"')
    If(itest = 2) then Do
      say "exe: " ear.x.jk "value is:" eal.x.jk
    End /* If(itest = 2) then Do */
  End /* Do jk = 1 to je */
End /* Do jk = 1 to jv */

/**************************************************
/* Now match up the exec vars with the proc vars. */
/**************************************************

jt = Ø
memtot. = ''
If(itest = 2) then Do
  say 'The number of procedures is' ip
  say 'The number of exec stats is' ie
End /* If(itest = 2) then Do */

/* ds.proc#.ds# */

Do jk1 = 1 to ip
  If(itest = 2) then Do
    say 'Am processing procedure:' jk1
  End /* If(itest = 2) then Do */
  iflag3 = Ø
  pnum = Ø
  enum = Ø
  Do jk = 1 to ie while iflag3 = Ø
    If(execnam.jk = procnam.jk1) Then Do
      pnum = jk1
      enum = jk
      iflag3 = 1
    End /* If(execnam.jk = procnam.jk1) Then Do */
    If(itest = 2) then Do
      say 'pnum is' pnum 'enum is' enum
    End /* If(itest = 2) then Do */
End /* If(execnam.jk = procnam.jk1) Then Do */
End /* Do jk = 1 to ie */
If(itest = 2 & iflag3 = Ø) then Do
  say 'No match found for procnam:' procnam.jk1
End /* If(itest = 2) then Do */
Do jk = 1 to dn.jk1
  n = Ø
  Do until ds.jk1.jk = ''
    n = n + 1
    parse var ds.jk1.jk mem.jk1.jk.n '.' ds.jk1.jk
    If(mem.jk1.jk.n = '') then Do
      n = n - 1
    End /* If(mem.jk1.jk.n = '') then Do */
  End /* Do until ds.ip.jk = '' */
  jt = jt + 1
Do jn = 1 to n
  If(substr(mem.jk1.jk.jn,1,1) = '&') then Do
    temp1 = substr(mem.jk1.jk.jn,2)
    iflag1 = Ø
    Do xx = 1 to parn.pnum while iflag1 = Ø
      If(temp1 = par.pnum.xx) then Do
        mem.jk1.jk.jn = pal.pnum.xx
        Do yy = 1 to earn.enum
          If(temp1 = ear.enum.yy) then Do
            mem.jk1.jk.jn = eal.enum.yy
            iflag1 = 1
          End /* If(temp1 = ear.yy) then Do */
        End /* Do yy = 1 to earn.enum */
        iflag1 = 1
      End /* If(temp1 = par.xx) then Do */
    End /* Do xx = 1 to parn.pnum while iflag1 = Ø */
  If(memtot.jt = '') then Do
    memtot.jt = mem.jk1.jk.jn
  End /* If(memtot.jt = '') then Do */
If(memtot.jt = '') then Do
  memtot.jt = memtot.jt 33 '.' 33 mem.jk1.jk.jn
End /* If(memtot.jt = '') then Do */
Else Do
  memtot.jt = memtot.jt 33 '.' 33 mem.jk1.jk.jn
End /* If(memtot.jt = '') then Do */
End /* Do jn = 1 to n */
End /* Do jk = 1 to jd */
memtot.Ø = jt

/* This is what we want to end up with */
/* 'ISREDIT NULLS = ON' */
/* 'ISREDIT LINE_AFTER .ZCSR = " This is insert row " ' */
memh.1   = "//CHECKDD EXEC PGM=IEFBR14"
val1 = "'ISREDIT LINE_AFTER .ZCSR ="
val2 = memh.1
interpret val1 33 ' ' 33 val2 33 ' ' 33 ""
coli = Ø
rowi = rowi + 1
/* Find the max length of all dataset names. */
lends = Ø /* max length of all the datasets. */
Do jk = 1 to memtot.Ø
   memtot.jk = strip(memtot.jk,B,' ') '
   xx = length(memtot.jk)
   If(xx > lends) then Do
      lends = xx
   End /* If(xx > lends) then Do */
End /* Do jk = 1 to memtot.Ø */
/* The 26 is the length of the //STPnnn DD DSN= + 1 bit */
lends = lends + 26
Do jk = 1 to memtot.Ø
   /* Check if each dataset exists */
   xx = SYSDSN(memtot.jk)
   If(xx = 'OK') then Do
      eorn = ' ' 
   End /* If(xx = 'OK') then Do */
   Else Do
      eorn = '<=== DOES NOT EXIST ==='
   End /* If(xx = 'OK') then Do */
   memtot.jk = "//STP" 33 right(jk,3,'Ø') 33 " " 33 , "DD DISP=SHR,DSN=" 33 memtot.jk
   xx = length(memtot.jk)
   pad1 = lends - xx
   memtot.jk = memtot.jk 33 copies(' ',pad1) 33 eorn 'ISREDIT CURSOR = (ROWI,COLI)'
val1 = "'ISREDIT LINE_AFTER .ZCSR ="
val2 = memtot.jk
interpret val1 33 ' ' 33 val2 33 ' ' 33 ""
rowi = rowi + 1
End /* Do jk = 1= to nisr */
'ISREDIT CURSOR = (ROWI,COLI)'

© Xephon 1998
Programart has announced Version 2.0 of its Strobe MVS for Sysplex, as well as Version 4.0 of its APMpower. The new releases include Y2K compliance and support for international date and time formats. Additional enhancements for the two products include support for the most recent versions of language compilers and subsystems including CICS Transaction Server for OS/390 Release 1.2, OS/390 Version 2.4, DB2 Version 5, COOL:Gen (Composer/IEF) Release 4.1a, CA-IDMS Release 14, ADABAS 6.1.3, IBM COBOL Version 2.1, PL/I Version 1.8, IBM C/C++ Version 3.5, and Language Environment V1R8. There’s also support for IBM’s BatchPipes for MVS (part of SmartBatch for OS/390), aimed at improving the performance of applications that use BatchPipes. What’s more, the CA-IDMS feature has been improved to make it easier to identify inefficient CA-IDMS transactions and ADS/O dialogues. Improvements specific to APMpower include more extensive help for analyzing COOL:Gen application performance. Also, users of SQL Server 6.5 and Oracle 7.3 for NT can now use and configure APMpower’s profile library. Strobe starts at $10,995, and is based on the processing capacity of the machine. A single-user licence of APMpower is $2,000, while a concurrent-use licence costs $4,000.

For further information contact: Programart Corp, 124 Mount Auburn Street, University Place, Cambridge, MA 02138, USA.
Tel: (617) 661 3020
Fax: (617) 498 4010

IBM has announced Version 1.2 of its Maintenance 2000 tool for analysing MVS programs in OS/390 environments. It statically analyses PL/I source programs, %INCLUDE statement and macros, COBOL source programs and copybooks, CA-Easytrieve Plus programs and macros, and JCL and catalogued procedures. The software provides an impact analysis that focuses on data flow, generates a cross-reference list for programs, jobs, copybook (%INCLUDE) files, and datasets, supports both batch and on-line applications, works on DB2, CICS, and DL/I applications, and searches for two-digit date items for system-wide impact analysis. IBM plans to enhance the product to support COBOL MLE feature and to support the future releases of OS/390 COBOL and MVS COBOL. One-off licence charges start at $37,500.

IBM has also announced Net.Commerce Pro, aimed at large companies, which includes the same as the Start programme, plus catalog tools and back-end integration tools for accessing software such as CICS, MQSeries, IMS, SAP R/3, and EDI. It runs on OS/390, AIX, NT, and Solaris. Meanwhile, IBM extended the platform support of its CommercePoint payment products, including making the CommercePoint eTill available on OS/390 and Solaris. Also new is CommercePoint Gateway for OS/390, providing an interface between merchant Web servers and existing credit card processing systems.

Contact your local IBM representative for further information.