142 MVS update

July 1998

In this issue

Published by
Xephon
27-35 London Road
Newbury
Berkshire RG14 1JL
England
Telephone: 01635 33598
From USA: 01144 1635 33598
E-mail: xephon@compuserve.com

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

Contributions
If you have anything original to say about MVS, or any interesting experience to recount, why not spend an hour or two putting it on paper? The article need not be very long – two or three paragraphs could be sufficient. Not only will you be actively helping the free exchange of information, which benefits all MVS users, but you will also gain professional recognition for your expertise, and the expertise of your colleagues, as well as some material reward in the form of a publication fee – we pay at the rate of £170 ($250) per 1000 words for all original material published in *MVS Update*. If you would like to know a bit more before starting on an article, write to us at one of the above addresses, and we'll send you full details, without any obligation on your part.

Editor
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.

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.

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.

Printed in England.

Displaying SYS1.BRODCAST message area users

THE PROBLEM

The SYS1.BRODCAST message area (the area where TSO SEND messages or the NOTIFY message for a batch job are sent) at our site fills up approximately annually. Generally this is because messages are being sent to dormant, inactive, or suspended TSO user-ids and usually there is no easy way of identifying who the big offenders are.

THE SOLUTION

The BRODMAIL program described by this article can be used to display the user-ids in SYS1.BRODCAST that are using more than a threshold number of brodcast message entries. The default threshold has been set at 25, however you can customize your request by passing a parameter (a numeric value between 1 and 9999) to the program.

You can invoke the program by doing a CALL command from either the READY prompt or from the command prompt of ISPF option 6. For example, if you issue the command:

```
CALL 'my.linklib(BRODMAIL)' '40'
```

you will get a list of user-ids that have 40 or more unread messages in SYS1.BRODCAST. This generates output of the following format:

```
USERIDS WITH 40 OR MORE UNREAD BRODCAST MESSAGES.

TSO25        309
TSO10        131
PRDJOB1       48
TSO47         40
PRDUSR9       46
BACKUP        65
HSMCLN       104
***
```

The obvious question now is ‘how do I get things cleaned up?’. One of the following three options could be applied:

1 You get the offending user to log in to TSO. This will flush the saved messages from SYS1.BRODCAST.

2 You log in to TSO under the offending users-id. This will flush the saved messages from SYS1.BRODCAST.

3 You use a utility presented in Issue 26 of *MVS Update* (November, 1988). The article – *Reading other users TSO mail* – describes a command that can be used to purge the TSO messages for specified user-ids.

The BRODMAIL program has no special linkage editor requirements and requires that you have a minimum of READ access authority to SYS1.BRODCAST.

BRODMAIL

```
         MACRO
&LBL     @AMODE &MODE
.*   THE CONTENTS OF REG R14 ARE NOT PRESERVED BY THIS MACRO
.*   MODE:  IS THE ADDRESSING MODE YOU WANT TO SET
         AIF   ('&MODE' EQ '24').AMODE24
         AIF   ('&MODE' EQ '31').AMODE31
         MNOTE 5,'INVALID MODE SPECIFIED'
         AGO   .DONE                   ALL DONE
.AMODE24 ANOP
&LBL     LA    14,*+6                  SET ADDRESS VALUE
         BSM   Ø,14                    SET ADDRESS MODE
         AGO   .DONE                   ALL DONE
.AMODE31 CNOP  Ø,4                     ALIGN TO FULL WORD
&LBL     L     14,*+8                  SET ADDRESS VALUE
         B     *+8                     BRANCH AROUND CONSTANT
         DC    A(X'8ØØØØØØØ'+*+6)      BRANCH ADDRESS AND ADDRESS MODE
         BSM   Ø,14                    SET ADDRESS MODE
.DONE    ANOP
         MEND
*********************************************************************
*   THE FOLLOWING PROGRAM IS USED TO EXAMINE THE MESSAGE SECTION OF   *
*   SYS1.BRODCAST AND RETURN A LIST OF USER-IDS THAT HAVE MORE        *
*   UNREAD MESSAGES IN SYS1.BRODCAST THAN SPECIFIED IN THE MSGMIN     *
*   PROGRAM PARAMETER (DEFAULT IS SET AT 25).  THIS CAN BE A USEFUL   *
*   TOOL WHEN THE MESSAGE AREA OF THE BRODCAST DATASET HAS FILLED     *
*   AND YOU ARE TRYING TO DETERMINE IF THERE ARE ANY HEAVY USERS OF   *
*   THE BRODCAST MESSAGE AREA.  THIS PROGRAM, WHEN USED IN            *
*   CONJUNCTION WITH THE LISTBCX PROGRAM FROM MVS UPDATE ARTICLE      *
*   READING OTHER USERS TSO MAIL (FROM MVS UPDATE ISSUE 26, NOV 1988),*
*   CAN BE AN EFFECTIVE TOOL IN QUICKLY FREEING SPACE IN A FULL       *
*   SYS1.BRODCAST DATASET SITUATION.                                  *
*                                                                     *
*   THE PROGRAM CAN BE EXECUTED AS FOLLOWS:                           *
*      CALL 'MY.LINKLIB(BRODMAIL)' '5Ø'                               *
```

```
*                                                                     *
*   THE ABOVE INVOCATION WOULD RETURN A LIST OF USER-IDS WHO HAVE 5Ø  *
*   OR MORE UNREAD MESSAGES IN SYS1.BRODCAST.                         *
***********************************************************************
*
BRODMAIL CSECT
BRODMAIL AMODE 31
BRODMAIL RMODE 24
         PRINT NOGEN
         STM   R14,R12,12(R13)         SAVE THE ENVIRONMENT
         LR    R1Ø,R15                 SET MODULE ...
         USING BRODMAIL,R1Ø            ADDRESSABILITY
         ST    R13,SAVEAREA+4          SAVE OLD SAVEAREA
         LA    R13,SAVEAREA            GET NEW SAVEAREA
*   EXTRACT INCOMING MSGMIN PARM
         L     R2,Ø(,R1)               GET PARM ADDRESS
         XR    R3,R3                   CLEAR R3
         LH    R3,Ø(,R2)               GET PARM LENGTH
         LTR   R3,R3                   ZERO?
         BZ    ALLOCIT                 YES - USE THE DEFAULT
         CH    R3,=H'4'                IS THE LENGTH TOO BIG?
         BH    PARMERR                 YES - ISSUE A MESSAGE
         LA    R2,2(,R2)               GET PARM ADDRESS
         LR    R15,R2                  SAVE FOR OTHER USES
         LR    R14,R3                  SAVE THE LENGTH VALUE AS WELL
PARMLOOP TM    Ø(R15),X'FØ'            A DECIMAL DIGIT?
         BNO   PARMERR                 NO - ISSUE A MESSAGE
         LA    R15,1(,R15)             POINT TO NEXT PARM BYTE
         BCT   R14,PARMLOOP            CHECK IT OUT
         BCTR  R3,Ø                    REDUCE PARM LENGTH BY ONE
         MVC   DBL1(8),=8C' '          PRIME THE WORK AREA
         EX    R3,PARMMVC              MOVE THE PARM
         MVC   HEADER+13(4),=8C' '     CLEAR THE HEADER AREA
         LA    R14,DBL1(R3)            POINT TO LAST DIGIT
         LA    R15,HEADER+16           POINT TO HEADER AREA
         LA    R1,1(,R3)               SET THE PROPER LOOP COUNT
LIMVAL   MVC   Ø(1,R15),Ø(R14)         MOVE THE DIGIT
         BCTR  R14,Ø                   REDUCE POINTER BY ONE
         BCTR  R15,Ø                   REDUCE POINTER BY ONE
         BCT   R1,LIMVAL               IF MORE, LOOP BACK
         EX    R3,PARMTR               TRANSLATE THE PARM
         ST    R3,PARMLEN              SAVE THE PARM LENGTH
         OI    PARMLEN+3,X'7Ø'         SET LENGTH VALUE FOR EX
         L     R15,PARMLEN             LOAD THE LENGTH VALUE FOR EX
         EX    R15,PARMPACK            PACK THE INCOMING PARM VALUE
         OI    DBL2+7,X'ØC'            SET A SIGN
         CVB   R2,DBL2                 CONVERT TO BINARY
         ST    R2,MSGMIN               SAVE THE MSGMIN VALUE
*
*   ALLOCATE 'SYS1.BRODCAST' DATASET
```

```
ALLOCIT  MVI   S99VERB,ALLOC           SET ALLOCATE FLAG
         MVC   S99FLAG1(4),=F'Ø'      CLEAR STATUS
         MVC   S99INFO(2),=F'Ø'       CLEAR EXTRA INFO
         LA    R1,S99RBPTR            LOAD PARAMETER ADDRESS
         SVC   99                     ALLOCATE
         LTR   R15,R15                ALLOCATED?
         BNZ   ALLOCERR               NO - ISSUE AN ERROR MESSAGE
OPENIT   OPEN  (BRODCAST,(INPUT))     OPEN FILE
         LA    R2,BRODCAST            ADDRESS DCB
         USING IHADCB,R2
         TM    DCBOFLGS,X'1Ø'         OPEN OK?
         BNZ   GETREC1                YES - CONTINUE
         B     UNALLOC                GO DEALLOCATE
*   READ THE FIRST RECORD AND DETERMINE IF THE BRODCAST DATASET IS
*   INITIALIZED.  IF IT IS, GET THE RBA OF THE NOTICE SECTION.
GETREC1  MVC   RBA,=F'Ø'              SET RBA VALUE
         BAL   R14,READ               READ RECORD
         LA    R2,RECORD              ADDRESS FIRST RECORD
         USING BRODREC1,R2
         XC    DIRBASE(4),DIRBASE     CLEAR AREA
         XC    NOTICCNT(4),NOTICCNT   CLEAR AREA
         CLC   BRODDSN,=C' SYS1.BRODCAST DATA SET ' INITIALIZED?
         BNE   NOTINIT                NO - ERROR
         CLC   BRODLVL,=C'LEVEL 2'    INITIALIZED?
         BE    INITOK                 YES - CONTINUE
NOTINIT  EQU   *
         B     CLOSEIT                GO CLOSE BRODCAST
INITOK   MVC   DIRRBA,RBAMAIL         GET RBA OF MAIL RECORD
         TPUT  HEADER,HDRLEN          WRITE OUT A HEADER MESSAGE
         TPUT  OUTMSG,L'OUTMSG        WRITE A BLANK LINE
MAILDIR  MVC   RBA,DIRRBA             GET RBA OF MAIL DIRECTORY
         CLC   RBA(3),=3X'ØØ'         END OF MAIL DIRECTORY?
         BNE   READMAIL               NO - GO READ MAIL
         B     CLOSEIT                ALL DONE
READMAIL BAL   R14,READ               BRING IN RECORD
         MVC   RECSAVE(129),RECORD    SAVE THE MAIL DIRECTORY RECORD
         USING MAIL,R6
         LA    R6,RECSAVE             GET MAIL DIRECTORY RECORD ADDR
         LR    R8,R6                  SAVE RECORD ADDRESS
         LA    R8,129(,R6)            GET ADDRESS OF END OF RECORD
         L     R9,=F'9'               LOAD LOOP COUNTER
USERLOOP EQU   *
         CLC   Ø(7,R6),=7X'ØØ'        A BLANK ENTRY?
         BE    GETNEXT                YES - GO GET THE NEXT ONE
         CLC   7(3,R6),=7X'ØØ'        ANY MAIL MESSAGES?
         BE    GETNEXT                NO - GO GET THE NEXT ONE
         XR    R5,R5                  CLEAR MESSAGE COUNTER
         MVC   SAVEUSER(7),Ø(R6)      SAVE THE USER ID
         MVC   RBA(3),7(R6)           GET RBA OF MAIL MESSAGE
NEXTMAIL BAL   R14,READ               READ THE MESSAGE
```

```
         LA    R5,1(,R5)               ADD ONE TO COUNTER
         MVC   RBA(3),RECORD+126       GET RBA OF NEXT MAIL MESSAGE
         CLC   RBA(3),=7X'ØØ'          ANY MORE?
         BE    PUTINFO                 NO - WRITE THE INFO
         B     NEXTMAIL                READ NEXT MAIL MESSAGE
GETNEXT  EQU   *
         LA    R6,ENTLEN(,R6)          POINT TO NEXT USER ENTRY
         BCT   R9,USERLOOP             CHECK ALL OVER
         CLC   RECSAVE+126(3),=7X'ØØ'  ALL DONE WITH MAIL DIRECTORY?
         BE    CLOSEIT                 YES - ALL DONE
         MVC   RBA(3),RECSAVE+126      GET NEXT MAIL DIRECTORY RBA
         B     READMAIL                READ NEXT MAIL DIRECTORY RECORD
PUTINFO  EQU   *
         C     R5,MSGMIN               IS IT HIGH ENOUGH TO WORRY?
         BL    GETNEXT                 NO - DON'T EVEN MENTION
         LR    R3,R5                   SAVE MESSAGE COUNT
         CVD   R3,DBL1                 CONVERT IT
         UNPK  DBL2(8),DBL1+4(4)       UNPACK IT
         OI    DBL2+7,X'FØ'            CHANGE SIGN
         L     R15,=F'8'               SET LOOP COUNTER
         LA    R14,DBL2                GET FIELD ADDRESS
MOVEPAD  CLI   Ø(R14),C'Ø'             IS IT A PAD CHARACTER?
         BNE   DONEPAD                 NO - WE'RE ALL DONE
         MVI   Ø(R14),C' '             REPLACE WITH A BLANK
         LA    R14,1(,R14)             POINT TO NEXT BYTE
         BCT   R15,MOVEPAD
DONEPAD  EQU   *
         MVC   OUTMSG(7),SAVEUSER      MOVE IN THE USER NAME
         MVC   OUTMSG+8(8),DBL2        MOVE IN THE MESSAGE COUNT
         TPUT  OUTMSG,L'OUTMSG         WRITE THE INFO
         B     GETNEXT                 GET NEXT ENTRY
CLOSEIT  CLOSE (BRODCAST)              CLOSE FILE
*   DEALLOCATE DATASET
UNALLOC  MVI   S99VERB,DEALLOC         SET DEALLOCATE FLAG
         MVC   S99FLAG1(4),=F'Ø'       CLEAR STATUS
         MVC   S99INFO(2),=F'Ø'        CLEAR EXTRA INFO
         OI    S99TUPL,X'8Ø'           ONLY DDNAME REQUIRED
         LA    R1,S99RBPTR             LOAD PARAMETER ADDRESS
         SVC   99                      GO DEALLOCATE
RETURN   XR    R15,R15                 SET RETURN CODE
         L     R13,SAVEAREA+4          GET SAVEAREA ADDRESS
         L     R14,12(R13)             GET RETURN ADDRESS
         LM    RØ,R12,2Ø(R13)          RESTORE THE ENVIRONMENT
         BR    R14                     RETURN
READ     EQU   *
         ST    R14,R14SAVE             SAVE RETURN ADDRESS
         XC    FEEDBACK,FEEDBACK       CLEAR FEED BACK AREA
         MVC   FEEDBACK(3),RBA         SET RBA
        @AMODE 24
         READ  DECBINØ1,DIF,BRODCAST,RECORD,'S',KEY,FEEDBACK
```

```
         CHECK DECBINØ1                WAIT FOR IO
         @AMODE 31
         L     R14,R14SAVE             GET RETURN ADDRESS
         BR    R14                     RETURN
PARMERR  EQU   *
         TPUT  ERRMSGØ,LENEMSGØ        WRITE OUT AN ERROR MESSAGE
         B     RETURN                  JUST RETURN
ALLOCERR EQU   *
         TPUT  ERRMSG1,LENEMSG1        WRITE OUT AN ERROR MESSAGE
         B     RETURN                  JUST RETURN
SYNADEX1 EQU   *
         TPUT  ERRMSG2,LENEMSG2        WRITE OUT AN ERROR MESSAGE
         B     CLOSEIT                 CLOSE DATASET AND RETURN
ERRMSGØ  DC    C'INVALID MSGMIN PARM VALUE ENTERED.  '
         DC    C'OPERATION TERMINATED.'
LENEMSGØ EQU   *-ERRMSGØ
ERRMSG1  DC    C'BRODCAST DATASET CAN NOT BE ALLOCATED.  '
         DC    C'OPERATION TERMINATED.'
LENEMSG1 EQU   *-ERRMSG1
ERRMSG2  DC    C'PROBLEM PROCESSING BRODCAST DATASET.  '
         DC    C'OPERATION TERMINATED.'
LENEMSG2 EQU   *-ERRMSG2
HEADER   DC    C'USERIDS WITH   25 OR MORE UNREAD BRODCAST MESSAGES.'
HDRLEN   EQU   *-HEADER
MSGMIN   DC    F'25'                   SET DEFAULT # OF MSGS/USERID
SYSIKJBC DC    C'SYSIKJBC'
MSGMVC   MVC   MSGTEXT(1),RECORD+1
PARMMVC  MVC   DBL1(1),Ø(R2)
PARMTR   TR    DBL1(1),TRTABLE
PARMPACK PACK  DBL2(1),DBL1(1)
MSG#     DS    CL3
MSGTEXT  DS    CL125
         DC    C' '
SAVEAREA DS    18F
DBL1     DS    2D
DBL2     DS    2D
R14SAVE  DS    F
PARMLEN  DS    F
NOTICCNT DC    F'Ø'
DIRBASE  DC    F'Ø'
EDITID   DC    C'BRODCAST.NOTICES  '
TRTABLE  DC    256X'8Ø'
         ORG   TRTABLE+Ø
         DC    C'Ø123456789ABCDEF'
         ORG   TRTABLE+193
         DC    X'ØAØBØCØDØEØF'
         ORG   TRTABLE+24Ø
         DC    X'ØØØ1Ø2Ø3Ø4Ø5Ø6Ø7Ø8Ø9'
         ORG   ,
OUTMSG   DC    CL16' '
```

```
NUMBER   DS    CL2                     MESSAGE NUMBER
RECORD   DS    CL129
RECSAVE  DS    CL129
SAVEUSER DS    CL7
FEEDBACK DS    CL8
DIRRBA   DS    X'000000'
RBA      DS    X'000000'
KEY      DS    X'00'
BRODCAST DCB   DSORG=DA,                                                X
               MACRF=RIC,                                              X
               RECFM=F,                                                X
               DDNAME=SYSLBC01,                                        X
               OPTCD=R,                                                X
               BUFNO=255,                                              X
               SYNAD=SYNADEX1
*   DYNAMIC ALLOCATE PARAMETER AREA
S99RBPTR DC    X'80',AL3(S99RB)        ADDRESS OF PARM LIST
S99RB    DC    X'14'
S99VERB  DC    X'01'
ALLOC    EQU   X'01'
DEALLOC  EQU   X'02'
S99FLAG1 DC    X'0000'
S99ERROR DC    X'0000'
S99INFO  DC    X'0000'
S99TXTP  DC    AL4(S99TUPL)
         DC    F'0'
S99FLAG2 DC    F'0'
S99TUPL  DC    AL4(TU0001)
TU2      DC    AL4(TU0002)
TU3      DC    AL4(TU0003)
TU4      DC    AL4(TU0004)
TU5      DC    X'80',AL3(TU0005)
* //TU0001  DD DSN=TU0002,DISP=(TU0003,TU0004,TU0005)
TU0001   DC    X'0001',X'0001',X'0008'
DDNAME   DC    CL8'SYSLBC01'                  //SYSLBC01 DD
TU0002   DC    X'0002',X'0001',X'002C'
DSN      DC    CL44'SYS1.BRODCAST'            DSN=SYS1.BRODCAST,
TU0003   DC    X'0004',X'0001',X'0001',X'08'  DISP=(SHR,
TU0004   DC    X'0005',X'0001',X'0001',X'08'            KEEP,
TU0005   DC    X'0006',X'0001',X'0001',X'08'                 KEEP)
         LTORG
         $REQU
         DCBD  DSORG=DA
BRODREC1 DSECT
         DS    C
RBANOTIC DS    CL3                     RBA OF NOTICES DIRECTORY
         DS    C
RBAMAIL  DS    CL3                     RBA OF MAIL DIRECTORY
RECTOTAL DS    F                       TOTAL RECORDS
MSGMAX   DS    H                       MAX. NUMBER OF MESSAGES
```

```
BRODDSN  DS    CL24                    DATASET NAME
BRODLVL  DS    CL7                     LEVEL
NOTICES  DSECT
NOTICFLG DS    CL1
NOTICFRE EQU   X'8Ø'                   NOTICE NOT IN USE
NOTICEND EQU   X'7F'                   END OF NOTICE DIRECTORY ENTRY
NOTICE#  DS    CL1
NOTICRBA DS    CL3
MAIL     DSECT
USER1    DS    CL7
U1MSG1ST DS    CL3
U1MSGEND DS    CL3
ENTLEN   EQU   *-USER1
USER2    DS    CL7
U2MSG1ST DS    CL3
U2MSGEND DS    CL3
USER3    DS    CL7
U3MSG1ST DS    CL3
U3MSGEND DS    CL3
USER4    DS    CL7
U4MSG1ST DS    CL3
U4MSGEND DS    CL3
USER5    DS    CL7
U5MSG1ST DS    CL3
U5MSGEND DS    CL3
USER6    DS    CL7
U6MSG1ST DS    CL3
U6MSGEND DS    CL3
USER7    DS    CL7
U7MSG1ST DS    CL3
U7MSGEND DS    CL3
USER8    DS    CL7
U8MSG1ST DS    CL3
U8MSGEND DS    CL3
USER9    DS    CL7
U9MSG1ST DS    CL3
U9MSGEND DS    CL3
         DS    CL9
NEXTMRBA DS    CL3
         END
```

Jim Lautner
Systems Programmer (Canada)

Cleaning up datasets for deleted users

THE PROBLEM

A problem my installation has always had when deleting TSO user-ids is what to do about their datasets. When a user-id is deleted, it would be useful to be able to delete its UADS entry, catalog alias, and all of its associated datasets. Typically, however, other people in the group that the departing user-id belongs to need to get to some of the datasets that belong to the deleted user-id, often weeks or months afterwards. Having to keep the datasets around also means having to keep the catalog alias. Additionally, no one could keep track of the datasets left to go back and clean them up after some reasonable amount of time. Although we use DFSMShsm to migrate datasets after an appropriate level of inactivity, the datasets still remain cataloged under the user-id's high-level qualifier. We needed a way to be able to perform all the clean-up necessary to remove the userid from the system at one time, while being able to access the deleted user-id's datasets for some time after that.

THE SOLUTION

We came up with a relatively simple renaming scheme, which quickly blossomed into a bigger project. The result of that project was a REXX EXEC called CLEANUP, which automates just about all of the work in removing all the catalog entries associated with the deleted user-id by deleting unnecesary datasets and renaming the rest by prefixing the entire dataset name with a new, single-character, high-level qualifier.

CLEANUP works by using a few IDCAMS LISTCAT commands to get some information about what catalog the user alias resides in as well as the names of all datasets catalogued under it. One of the problems we had was that we had more than one user catalog for our TSO users. The problem that this presents is that VSAM datasets whose entries are in one user catalog cannot be renamed to a new high-level qualifier that resides in another user catalog. To fix this, we decided to have more than one new high-level qualifier to rename datasets to; these are specified within a REXX SELECT group under the variable name OLDCAT. If you have multiple TSO user catalogs,

code would need to be added to check for each catalog name and assign a new high level as appropriate.

Since we use DFSMShsm, it was also decided that if a dataset for the deleted user-id was already migrated, we would use the HDELETE command to delete the migrated datasets from the migration volumes, rather than recall the datasets to perform a rename. You might want to change this processing to suit your needs. We also decided arbitrarily to delete the user ISPF profile and broadcast datasets, rather than rename them. Another area requiring modification is the names of the master catalogs from which to delete the alias (both the names and the number at your installation); we have three LPARs which share all DASD and catalogs, hence the three DELETE ALIAS statements that are generated.

The EXEC handles the removal/rename of non-VSAM, generation dataset, VSAM cluster/data/index, GDG base, and alias entries from the catalog. The GDG base entry deletions are generated after all dataset rename/deletes, and the catalog alias deletes are the last generated, all to preserve proper order. As a consequence of the GDG handling, GDG datasets are renamed to plain non-VSAM names that end in GnnnnVnn, but are not really GDG datasets; therefore the rebuilding of the GDG bases is necessary to rebuild the true GDG structure if the GDG datasets need to be brought back to their original form.

CLEANUP does some basic verifications prior to performing any actions. It checks that the prefix provided to it is a catalog alias, that the alias has at least one associated dataset to operate on, that any dataset whose VOLSER is MIGRAT is processed via HDELETE rather than DELETE/RENAME/ALTER, that the dataset named being renamed to does not already exist, and that no dataset name exceeds 42 characters in length. Should any of these happen, CLEANUP will not perform any actual processing, in most cases setting an error flag, which is checked at the end of the routine.

In addition to providing the prefix to be operated on, a second parameter of DELETE needs to be supplied to have CLEANUP actually perform its work. If not supplied, it will only display the commands it would have issued. If the DELETE parameter is supplied, it will then check to see if any errors were encountered (via the error flag), and if so, will only produce its diagnostic messages.

CLEANUP

```
/* REXX */
arg prefix option
if length(prefix) = Ø then
   do
     say "Target userid missing"         /* send out error message   */
     err = 1
     exit                                /* get out now              */
   end
say "About to delete" prefix "related datasets."
x = outtrap("trap.","*")                 /* turn on output trapping  */
"LISTC ENT('"prefix"') ALL"              /* issue listc ent command  */
if rc ¬= Ø then
   do
     say prefix "does not have a catalog alias."
     err = 1
     exit
   end
"LISTC LVL('"prefix"') VOL"              /* issue listc lvl command  */
if rc ¬= Ø then
   do
     say prefix "has no datasets to operate on."
     err = 1
     exit
   end
x = outtrap("off")                       /* turn off output trapping */
profdsn = prefix'.ISPF.ISPPROF'          /* profile dataset name     */
broddsn = prefix'.BRODCAST.DATA'         /* broadcast dataset name   */
c = Ø                                    /* array index              */
d = Ø                                    /* array index              */
err = Ø
                                         /* process each name trapped*/
do i = 1 to trap.Ø
   w1 = word(trap.i,1)
   w2 = word(trap.i,2)
   w3 = word(trap.i,3)
   w4 = strip(w3)
   w5 = strip(word(trap.i,4))            /* for GDG BASE             */
                                         /* check LISTC types        */
   select
     when w1 = 'ALIAS' then              /* get alias catalog name   */
          do
            i = i + 5
            oldcat = substr(strip(trap.i),1Ø)
            select
              when oldcat = 'SYS1.ICFUCAT.VTSOØØ1' then
                   do
                     pref = 'X.'
                     iterate
                   end
              when oldcat = 'SYS1.ICFUCAT.VTSOØØ2' then
                   do
                     pref = 'Y.'
```

```
              iterate
            end
          otherwise
            do
              say "No corresponding rename prefix for" prefix ,
              "in catalog" oldcat
              err = 1
              exit
            end
        end
      end
when substr(w1,1,7) = 'VOLSER-' then
      do
        l = length(w1) - 6 + 1
        migrat = substr(w1,l,6)
        if migrat = 'MIGRAT' then
           cmd.c = 'HDEL ' word(cmd.c,2)
      end
when w1 = 'NONVSAM' then            /* non-VSAM file?           */
      do                            /* yes, process it          */
        err = chklen(w3)            /* check for dsn len error  */
        c = c + 1
        if   w3 = profdsn |,        /* is this a profile?       */
             w3 = broddsn then      /*         a broadcast?     */
             cmd.c = "DEL   '"w3"'" /* yes, delete it           */
        else cmd.c = "REN   '"||,   /* no, just do rename       */
               w3"' '"pref||w4"'"
      end
when w1 = 'CLUSTER' then            /* VSAM cluster?            */
      do                            /* yes, process it          */
        err = chklen(w3)            /* check for dsn len error  */
        c = c + 1
        cmd.c = "ALTER '"w3"' '"pref||w4"'"
      end
when w1 = 'DATA' then               /* VSAM data component?     */
      do                            /* yes, process it          */
        err = chklen(w3)            /* check for dsn len error  */
        c = c + 1
        cmd.c = "ALTER '"w3"' '"pref||w4"'"
      end
when w1 = 'INDEX' then              /* VSAM index component?    */
      do                            /* yes, process it          */
        err = chklen(w3)            /* check for dsn len error  */
        c = c + 1
        cmd.c = "ALTER '"w3"' '"pref||w4"'"
      end
when w1 = 'GDG' then                 /* GDG base?               */
      if w2 = 'BASE' then
      do                             /* yes, process it         */
        d = d + 1
        cmdg.d = "DEL   '"w5"' GDG"  /* delete it               */
      end
otherwise
```

```
          iterate
     end
end
                                          /* add GDG bases to array  */
do i = 1 to d
   c = c + 1
   cmd.c = cmdg.i
end
                                          /* add aliases to array    */
c = c + 1
cmd.c = "DEL   '"prefix"' ALIAS CAT('SYS1.ICFMCAT.VSYSØØ1')"
c = c + 1
cmd.c = "DEL   '"prefix"' ALIAS CAT('SYS1.ICFMCAT.VSYSØØ2')"
c = c + 1
cmd.c = "DEL   '"prefix"' ALIAS CAT('SYS1.ICFMCAT.VSYSØØ3')"
                                          /* check for dup datasets  */
do i = 1 to c
   x = substr(cmd.c,1,3)
   if x = 'REN' |  x = 'ALT' then
      do
        oldnm = word(cmd.i,2)
        newnm = word(cmd.i,3)
        If sysdsn(newnm) = 'OK' then
           do
             say "Cannot RENAME '"oldnm"' duplicate name found"
             err=1
             iterate
           end
      end
   if option = 'DELETE' then
      if err = Ø then
         interpret "cmd.i"
      else nop;
   else say cmd.i
end
exit
                                          /* check dsn lengths       */
chklen: procedure
arg dsn
if length(dsn) > 42 then
   do
     err = 1
     say "Dataset" dsn "name exceeds 42 characters."
   endarg prefix option
if length(prefix) = Ø then
   do
     say "Target userid missing"         /* send out error message   */
     err = 1
     exit                                /* get out now              */
   end
say "About to delete" prefix "related datasets."
x = outtrap("trap.","*")                  /* turn on output trapping  */
"LISTC ENT('"prefix"') ALL"               /* issue listc ent command  */
```

```
if rc ¬= Ø then
   do
     say prefix "does not have a catalog alias."
     err = 1
     exit
   end
"LISTC LVL('"prefix"') VOL"              /* issue listc lvl command  */
if rc ¬= Ø then
   do
     say prefix "has no datasets to operate on."
     err = 1
     exit
   end
x = outtrap("off")                       /* turn off output trapping */
profdsn = prefix'.ISPF.ISPPROF'          /* profile dataset name     */
broddsn = prefix'.BRODCAST.DATA'         /* broadcast dataset name   */
c = Ø                                    /* array index              */
d = Ø                                    /* array index              */
err = Ø
                                         /* process each name trapped*/
do i = 1 to trap.Ø
   w1 = word(trap.i,1)
   w2 = word(trap.i,2)
   w3 = word(trap.i,3)
   w4 = strip(w3)
   w5 = strip(word(trap.i,4))            /* for GDG BASE             */
                                         /* check LISTC types        */
   select
     when w1 = 'ALIAS' then              /* get alias catalog name   */
          do
            i = i + 5
            oldcat = substr(strip(trap.i),1Ø)
            select
              when oldcat = 'SYS1.ICFUCAT.VTSOØØ1' then
                   do
                     pref = 'X.'
                     iterate
                   end
              when oldcat = 'SYS1.ICFUCAT.VTSOØØ2' then
                   do
                     pref = 'Y.'
                     iterate
                   end
              otherwise
                   do
                     say "No corresponding rename prefix for" prefix ,
                     "in catalog" oldcat
                     err = 1
                     exit
                   end
            end
          end
     when substr(w1,1,7) = 'VOLSER-' then
```

```
          do
            l = length(w1) - 6 + 1
            migrat = substr(w1,l,6)
            if migrat = 'MIGRAT' then
               cmd.c = 'HDEL ' word(cmd.c,2)
          end
     when w1 = 'NONVSAM' then            /* non-VSAM file?           */
          do                             /* yes, process it          */
            err = chklen(w3)             /* check for dsn len error  */
            c = c + 1
            if   w3 = profdsn |,         /* is this a profile?       */
                 w3 = broddsn then       /*         a broadcast?     */
                 cmd.c = "DEL   '"w3"'"  /* yes, delete it           */
            else cmd.c = "REN   '"||,    /* no, just do rename       */
                    w3"' '"pref||w4"'"
          end
     when w1 = 'CLUSTER' then            /* VSAM cluster?            */
          do                             /* yes, process it          */
            err = chklen(w3)             /* check for dsn len error  */
            c = c + 1
            cmd.c = "ALTER '"w3"' '"pref||w4"'"
          end
     when w1 = 'DATA' then               /* VSAM data component?     */
          do                             /* yes, process it          */
            err = chklen(w3)             /* check for dsn len error  */
            c = c + 1
            cmd.c = "ALTER '"w3"' '"pref||w4"'"
          end
     when w1 = 'INDEX' then              /* VSAM index component?    */
          do                             /* yes, process it          */
            err = chklen(w3)             /* check for dsn len error  */
            c = c + 1
            cmd.c = "ALTER '"w3"' '"pref||w4"'"
          end
     when w1 = 'GDG' then                 /* GDG base?               */
          if w2 = 'BASE' then
          do                             /* yes, process it          */
            d = d + 1
            cmdg.d = "DEL   '"w5"' GDG"  /* delete it                */
          end
     otherwise
          iterate
     end
end
                                         /* add GDG bases to array   */
do i = 1 to d
   c = c + 1
   cmd.c = cmdg.i
end
                                         /* add aliases to array     */
c = c + 1
cmd.c = "DEL   '"prefix"' ALIAS CAT('SYS1.ICFMCAT.VSYSØØ1')"
c = c + 1
```

```
cmd.c = "DEL   '"prefix"' ALIAS CAT('SYS1.ICFMCAT.VSYSØØ2')"
c = c + 1
cmd.c = "DEL   '"prefix"' ALIAS CAT('SYS1.ICFMCAT.VSYSØØ3')"
                                          /* check for dup datasets  */
do i = 1 to c
   x = substr(cmd.c,1,3)
   if x = 'REN' |  x = 'ALT' then
      do
        oldnm = word(cmd.i,2)
        newnm = word(cmd.i,3)
        If sysdsn(newnm) = 'OK' then
           do
             say "Cannot RENAME '"oldnm"' duplicate name found"
             err=1
             iterate
           end
      end
   if option = 'DELETE' then
      if err = Ø then
         interpret "cmd.i"
      else nop;
   else say cmd.i
end
exit
                                          /* check dsn lengths       */
chklen: procedure
arg dsn
if length(dsn) > 42 then
   do
     err = 1
     say "Dataset" dsn "name exceeds 42 characters."
   end
return err
```


EXEC to find a member in a group of datasets

INTRODUCTION

This EXEC will search through a group of datasets for between one and 26 members. There are two ways to specify the datasets to be searched:

- In a member of a PDS
- As the result of a LISTCAT.

The EXEC has been designed to make specifying the search datasets as easy as possible. It is possible to use an asterisk as a wildcard in the search name (as shown in the help display). The EXEC then performs a LISTC LVL and ENT to pick up all the datasets required. It is worth experimenting with this to see what is returned for different invocations.

Two output sequential datasets are produced for a search:

- The first, called userid.FNDMEM.OUTPUT, contains the results of the search.
- The second, called userid.FNDMEM.OUTPUT2, contains a member list of every dataset searched.

Therefore, if you are looking for various members over the period of an install, you only have to run the EXEC once, and for subsequent searches you just have to browse the OUTPUT2 dataset. The OUTPUT and OUTPUT2 names can be changed at EXEC invocation time.

The EXEC help section can be displayed by issuing the command:

```
FINDMEM help
```

This explains provided detailed how to input the various parameters.

```
/* REXX */
/*********************************************************************/
/* To find a member in a group of datasets                           */
/*********************************************************************/
parse upper arg biglin
testflag = ' '
parse var biglin bigres '<' testflag
jk = 0
Do until testflag = ' '
  jk =jk + 1
  parse var testflag testf.jk testflag
End /* Do until testflag = ' ' */
jk1 = 0
Do while jk  > jk1
  jk1 = jk1 + 1
  If(testf.jk1 = 'TRACE') then Do
    trace r
  End /* If(testf.jk1 = 'TRACE') */
End /* Do jk1 = 1 to jk */
parse var bigres dsn1 whtlok norp excl .
If(pos(dsn1,'HELP') > 0 | dsn1 = '?' | dsn1 = ' ') then Do
  linp.1= "This EXEC is used to look at a group of datasets to find a"
  linp.2= "member in a particular dataset."
  linp.0 = 2
```

```
  indent = 1
  call format_text
  say "dsn1   - is the group stem ie OMEGA.IMS.YY.* (note,",
      "not YY*)."
  linp.1 = "You can also put a dataset(membername), which can"
  linp.2 = "contain a list of datasets to process.  An asterisk"
  linp.3 = "in col 1 of this member is treated as a comment."
  linp.0 = 3
  indent = 9
  call format_text
  say "whtlok - is the member(s) to be searched for sep by commas."
  say "norp   - whether to del/def the o/p file (NEW) or append (APP)",
  "- APP is the df"
  say "excl   - is the exclude list EXCL=dataset-name."
  linp.1 = "More than one datset can be specified, separated"
  linp.2 = "by a comma. These datasets will not be searched."
  linp.3 = "You can also use an asterisk as a"
  linp.4 = "wildcard."
  linp.0 = 4
  indent = 9
  call format_text
  say "The EXEC is invoked as ===> ",
      "FNDMEM  dsn1  whtlok  norp         "
  say "eg:                                "
  say "(a) FNDMEM  OMEGA.IMS.YY.*   KOISPAN   "
  say "(b) FNDMEM  OMEGA.IMS.YY.*   KOISPAN   NEW"
  say "(c) FNDMEM  OMEGA.IMS.YY.*   KOISPAN,FRED1   NEW"
  say "(d) FNDMEM  OMEGA.IMS.YY.*   KOISPAN,FRED1   APP"
  say "(e) FNDMEM  OMEGA.IMS.YY.*   KOISPAN,FRED*   NEW"
  say "(f) FNDMEM  OMEGA.IMS.YY.*   KOISPAN,*FRED*  NEW"
  say "You cannot do a FRE*D*D as a member name."
  say "You can specify up to 26 members to be searched for."
  say "(g) FNDMEM  OMEGA.IMS.YY.*   KOISPAN NEW EXCL=YOUR.DATASET.NM"
  say "(h) FNDMEM  OMEGA.IMS.YY.*   KOISPAN APP EXCL=YOUR.DATA*"
  ms1 = "Press PF1"
  ms2 = ,
  "FNDMEM dsn1 whtlok norp excl"
  ZEDSMSG = ms1
  ZEDLMSG = ms2
  "ISPEXEC SETMSG MSG(ISRZ000)"
  exit
End /* If(pos(dsn1,'HELP') > 0 | dsn1 = '?' | dsn1 = ' ') then Do */
/*                                                                 */
/* I had the following datasets: F1 = hlq.mlq.test1                */
/*                               F2 = hlq.mlq.test1.f2             */
/*                               F3 = hlq.mlq.test1.f3             */
/*                               F4 = hlq.mlq.test1.f4             */
/*                                                                 */
/* The table below shows what the LISTC LVL and ENT commands       */
/* return for differing dataset names used as input.               */
/*                                                                 */
/* |=======================================|                       */
```

```
/* |=d/s name=======|=== LVL ====|=== ENT ====|                    */
/* | hlq.mlq.test1  | F2 F3 F4   |    F1      |                    */
/* | hlq.mlq.test   | not listed | not listed |                    */
/* | hlq.mlq.test/  | invalid nm | invalid nm |                    */
/* | hlq.mlq.test*  | invalid nm | invalid nm |                    */
/* | hlq.mlq.test*. | invalid nm | invalid nm |                    */
/* |=========================================|                    */
/*                                                                 */
upper whtlok
upper dsn1
temp = whtlok
xx = Ø
Do until temp = ''
  xx = xx + 1
  parse var temp whtl.xx ',' temp
  wlet.xx = substr('ABCDEFGHIJKLMNOPQRSTUVWXYZ',xx,1)
End /* Do until temp = '' */
whtl.Ø = xx
Do jk3 = 1 to whtl.Ø
  If(pos('*',whtl.jk3) = Ø) then Do
    wtyn.jk3 = 1    /* FRED  */
  End /* If(pos('*',whtl.jk3) = Ø) then Do */
  Else Do
    If(substr(whtl.jk3,1,1) = '*') then Do
      wtyn.jk3 = 3  /* *FRED  */
    End /* If(substr(whtl.jk3,1,1) = '*') then Do */
    Else Do
      wtyn.jk3 = 2  /* FRED* */
    End /* If(substr(whtl.jk3,1,1) = '*') then Do */
  End /* If(pos('*',whtl.jk3) = Ø) then Do */
End /* Do jk3 = 1 to whtl.Ø */

iflag2 = Ø
If(pos('(',dsn1) > Ø) then Do
  /* member found */
  iflag2 = 1
  xx = outtrap('gvar.')
  Address TSO  "ALLOC FI(DD1) DA('"||dsn1||"') SHR"
  "EXECIO * DISKR DD1 (FINIS STEM DSP."
  Address TSO "FREE F(DD1)"
  xx = outtrap(OFF)
  yi = Ø
  Do jk7 = 1 to dsp.Ø
    If(substr(dsp.jk7,1,1) = '*') then Do
      Nop
    End /* If(substr(dsp.jk7,1,1) = '*') then Do */
    Else Do
      yi = yi + 1
      dsp.yi = subword(dsp.jk7,1,1)
    End /* If(substr(dsp.jk7,1,1) = '*') then Do */
  End /* Do jk7 = 1 to yy */
  nodsp = yi
```

```
  yy    = yi
End /* If(pos('(',dsn1) > Ø) then Do */

upper excl
If(excl = ' ') then Do
  xv = Ø
  excldsn = Ø
End /* If(excl = ' ') then Do */
Else Do
  parse var excl 'EXCL=' val
  xv = Ø
  Do until val = ''
    xv = xv + 1
    parse var val exn.xv ',' val
    If(pos('*',exn.xv) > Ø) then Do
      ext.xv = 1
    End /* If(pos('*',exn.xv) > Ø) then Do */
    Else Do
      ext.xv = Ø
    End /* If(pos('*',exn.xv) > Ø) then Do */
  End /* Do until val = '' */
End /* If(excl = ' ') then Do */

hed.1 =  'Looking in the following datasets:' dsn1
xh = 1
If(iflag2 = 1) then Do
  Do jk8 = 1 to yi
    xh = xh + 1
    hed.xh = dsp.jk8
  End /* Do jk8 = 1 to yi */
End /* If(iflag2 = 1) then Do */
xh = xh + 1
hed.xh = 'Looking for the following members:'
Do jk = 1 to whtl.Ø
  xh = xh + 1
  hed.xh = wlet.jk whtl.jk
End /* Do jk = 1 to whtl.Ø */
If(xv > Ø) then Do
  xh = xh + 1
  hed.xh = "The following datasets were excluded:"
  Do jk9 = 1 to xv
    xh = xh + 1
    hed.xh = exn.jk9
  End /* Do jk9 = 1 to xv */
End /* If(xv > Ø) then Do */

/* Check that the norp input parameter is acceptable. */
temp = norp
parse var temp norp '=' nam1
If(nam1 = ' ') then Do
  apnam1 = 'OUTPUT'
  apnam2 = 'OUTPUT2'
```

```
End /* If(nam1 = ' ') then Do */
Else Do
  If(length(nam1) > 8) then Do
    nam1 = substr(nam1,1,8)
  End /* If(length(nam1 > 8) then Do */
  If(length(nam1) > 7) then Do
    tt = substr(nam1,1,7)
  End /* If(length(nam1) > 7) then Do */
  Else Do
    tt = nam1
  End /* If(length(nam1) > 7) then Do */
  apnam1 = nam1
  apnam2 = tt || '2'
End /* If(nam1 = ' ') then Do */
If(apnam1 = apnam2) then Do
  say 'I cannot write out the 2 output files that I need to, because',
      'they both have'
  say 'the same llq:'
  say 'OUTPUT1 is:'apnam1
  say 'OUTPUT2 is:'apnam2
  say 'The OUTPUT2 name is the OUTPUT1 name with a 2 added.'
  say 'If OUTPUT1 is 8 characters, then OUTPUT2 is the first seven',
      'characters'
  say 'of OUTPUT1 with a 2 added.'
  say 'Therefore you cannot specify an 8 character name ending in a 2.'
  say 'and you specified ===>' temp
  exit
End /* If(apnam1 = apnam2) then Do */

/* End of norp check. */
xh = xh + 1
hed.xh = "Start Date/time:" DATE() "-" TIME()
xh = xh + 1
hed.xh = copies('=',79)
Do jk = 1 to xh
  say hed.jk
End /* Do jk = 1 to xh */
numl = Ø
nume = Ø
xu   = Ø
iali = Ø  /* The nos of datasets containing aliases. */
If(iflag2 = Ø) then Do
  /* no member list */
  xx = outtrap('gvar.')
  Address TSO  "LISTC LVL('" || dsn1 || "')"
  xx = outtrap(OFF)
  yy = Ø
  Do jk = 1 to gvar.Ø
    If(subword(gvar.jk,1,1) = 'NONVSAM') then Do
      yy = yy + 1
      dsp.yy = subword(gvar.jk,3,1)
      numl = numl + 1
```

```
    End /* If(subword(gvar.jk,1,1) = 'NONVSAM') then Do */
  End /* Do jk = 1 to gvar.Ø */
  xx = outtrap('gvar.')
  Address TSO  "LISTC ENT('" || dsn1 || "')"
  xx = outtrap(OFF)
  Do jk = 1 to gvar.Ø
    If(subword(gvar.jk,1,1) = 'NONVSAM') then Do
      yy = yy + 1
      dsp.yy = subword(gvar.jk,3,1)
      nume = nume + 1
    End /* If(subword(gvar.jk,1,1) = 'NONVSAM') then Do */
  End /* Do jk = 1 to gvar.Ø */
  If(yy = Ø) then Do
    outlin.1 = 'No matches found for the dataset name.'
    nodsp = 1
    say outlin.1
  End /* If(yy = Ø) then Do */
  Else Do
    xh = xh +1
    hed.xh = 'The number found thru LVL is:'numl
    say hed.xh
    xh = xh +1
    hed.xh = 'The number found thru ENT is:'nume
    say hed.xh
    nodsp = yy
  End /* If(yy = Ø) then Do */
End /* If(iflag2 = Ø) then Do */
If(yy = Ø) then Do
  Nop
End /* If(yy = Ø) then Do */
Else Do
  If(nodsp > 1Ø) then Do
    say 'There are ' nodsp 'datasets to process.  Do you want' ,
    'to continue (Y/n)?'
    parse pull ans
    upper ans
    If(ans = 'N') then Do
      say 'exiting ...'
      exit
    End /* If(ans = 'N') then Do */
  End /* If(nodsp > 1Ø) then Do */
  say "Do you want to see just the hits? (y/N)"
  parse pull ans
  upper ans
  Do jk = 1 to nodsp
    excldS = Ø
    If(xv > Ø) then Do
      Do jk4 = 1 to xv
        If(ext.jk4 = 1) then Do
          /* * exclusion */
          gg = length(exn.jk4)
          gg = gg - 1
```

```
        xx = substr(exn.jk4,1,gg)
        If(substr(dsp.jk,1,gg) = xx) then Do
          /* exclude the dataset */
          excldS = 1
        End /* If(substr(dsp.jk,1,gg) = xx) then Do */
      End /* If(ext.jk4 = 1) then Do */
      If(ext.jk4 = Ø) then Do
        ic = Ø
        gg = translate(exn.jk4,' ','. ')
        g1 = words(gg)
        hh = translate(dsp.jk,' ','. ')
        g2 = words(hh)
        Do jk5 = 1 to g1
          If(subword(gg,jk5,1) = subword(hh,jk5,1)) then Do
            ic = ic + 1
          End /* If(subword(gg,jk5,1) = subword(hh,jk5,1)) then */
        End /* Do jk5 = 1 to g1 */
        If(ic = g1) then Do
          excldS = 1
        End /* If(ic = g1) then Do */
      End /* If(ext.jk4 = Ø) then Do */
    End /* Do jk4 = 1 to xv */
  End /* If(xv > Ø) then Do */
  If(excldS = Ø) then Do
    xx = outtrap('gvar.')
    Address TSO  "LISTDS ('" || dsp.jk || "') MEMBERS"
    xx = outtrap(OFF)
    iflag1 = Ø
    Do jk1Ø = 1 to gvar.Ø while iflag1 = Ø
      If(subword(gvar.jk1Ø,1,1) = '—MEMBERS—') then Do
        iflag1 = 1
        is = jk1Ø + 1
      End /* If(subword(gvar.jk1Ø,1,1) = '—MEMBERS—') then Do */
    End /* Do jk1Ø = 1 to gvar.Ø while iflag1 = Ø */
    If(iflag1 = Ø) then Do
      /* No member list found - therefore a sequential file */
      is = Ø
    End /* If(iflag1 = Ø) then Do */
    xc = Ø
    hitl = ''
    forn = '<===='
    ihit = Ø
    If(is = Ø) then Do
      scr.jk = 'Sequential'
    End /* If(is = Ø) then Do */
    Else Do
      ifndali = Ø /* Does the ds contain aliases? Ø=N, 1=Y */
      Do jk1 = is to gvar.Ø
        xc = xc + 1
        wiw.jk.xc = subword(gvar.jk1,1,1)
        xu = xu + 1
        outlu.xu = left(dsp.jk,44,' ') || ' '||left(wiw.jk.xc,8,' ')
```

```
  If(words(gvar.jk1) = 2) then Do
   ifndali = 1
   parse var gvar.jk1 'ALIAS(' ali ')' .
   xc = xc + 1
   wiw.jk.xc = ali
   xu = xu + 1
   outlu.xu = left(dsp.jk,44,' ') || ' '||left(wiw.jk.xc,8,' ')
  End /* If(words(gvar.jk1 = 2) then Do */
End /* Do jk1 = is to gvar.Ø */
If(ifndali = 1) then Do
  iali = iali + 1
  fndali.iali = dsp.jk
End /* If(ifndali = 1) then Do */
xctot = xc
Do xc = 1 to xctot
  iage = Ø /* Have we passed the mem we are looking for Ø=N */
  Do jk3 = 1 to whtl.Ø
    If(wtyn.jk3 = 1) then Do
      If(whtl.jk3 = wiw.jk.xc) then Do
        hitl = hitl || wlet.jk3
        ihit = 1
      End /* If(whtl.jk3= wiw.jk.xc) then Do */
      If(wiw.jk.xc> whtl.jk3) then Do
        iage = 1
      End /* If(wiw.jk.xc > whtl.jk3) then Do */
    End /* If(wtyn.jk3 = 1) then Do */
    If(wtyn.jk3 = 2) then Do
      lw = length(whtl.jk3)
      lw = lw - 1
      xt = substr(whtl.jk3,1,lw)
      yt = substr(wiw.jk.xc,1,lw)
      If(xt = substr(wiw.jk.xc,1,lw)) then Do
        hitl = hitl || wlet.jk3
        ihit = 1
      End /* If(xt = substr(wiw.jk.xc,1,lw) then Do */
      If(yt > xt) then Do
        iage = 1
      End /* If(yt > xt) then Do */
    End /* If(wtyn.jk3 = 2) then Do */
    If(wtyn.jk3 = 3) then Do
      lw = length(whtl.jk3)
      lw = lw - 2
      xt = substr(whtl.jk3,2,lw)
      If(pos(xt,wiw.jk.xc) > Ø) then Do
        hitl = hitl || wlet.jk3
        ihit = 1
      End /* If(pos(xt,wiw.jk.xc) > Ø) then Do */
      /* We cannot check for iage, as we are looking for a */
      /* member with a wildcard at the start of the name.  */
    End /* If(wtyn.jk3 = 3) then Do */
  End /* Do jk3 = 1 to whtl.Ø */
  /* Only check if we can stop looking for the member if */
```

```
          /* the dataset does not contain alisases.              */
          If(ifndali = Ø) then Do
            If(iage = 1) then Do
              say 'Have trunc search of ' left(dsp.jk,44,' ') ,
              'at ' wiw.jk.xc
              leave xc
            End /* If(iage = 1) then Do */
          End /* If(ifndali = Ø) then Do */
        End /* Do xc = 1 to xctot */
        If(ihit = 1) then Do
          scr.jk = right(xctot,6,' ') || forn hitl
        End /* If(ihit = 1) then Do */
        Else Do
          scr.jk = right(xctot,6,' ')
        End /* If(ihit = 1) then Do */
      End /* If(is = Ø) then Do */
    End /* If(excldS = Ø) then Do */
    Else Do
      scr.jk = 'EXCLUDED'
    End /* If(excldS = Ø) then Do */
  End /* Do jk = 1 to nodsp */

  Do jk = 1 to nodsp
    outlin.jk = left(dsp.jk,44,' ') || ' ' || scr.jk
  End /* Do jk = 1 to nodsp */
  If(ans = 'Y') then Do
    Do jk = 1 to nodsp
      If(pos('===',scr.jk) > Ø) then Do
        say left(dsp.jk,44,' ') || ' ' || scr.jk
      End /* If(pos('===',scr.jk) > Ø) then Do */
    End /* Do jk = 1 to nodsp */
  End /* If(ans = 'Y') then Do */
  Else Do
    Do jk = 1 to nodsp
      say left(dsp.jk,44,' ') || ' ' || scr.jk
    End /* Do jk = 1 to nodsp */
  End /* If(ans = 'Y') then Do */
End /* If(yy = Ø) then Do */
If(iali >Ø) then Do
  xf = Ø
  xf = xf + 1
  xfh.xf = 'The following datasets contained aliases:'
  say xfh.xf
  Do jk = 1 to iali
    xf = xf + 1
    xfh.xf = fndali.jk
    say xfh.xf
  End /* Do jk = 1 to iali */
End /* If(ifndali >Ø) then Do */
/*************************************************************/
/* Write out the results                                     */
/*************************************************************/
```

```
tal.Ø = 1
tal.1 = copies('*',79)
dsn2 = userid() || '.FNDMEM.' || apnam1
dsn3 = userid() || '.FNDMEM.' || apnam2
/* Check if the o/p datasets exist. */
iext = Ø
xx = SYSDSN(dsn2)
If(xx = 'OK') then Do
  Nop
End /* If(xx = 'OK') then Do */
Else Do
  iext = 1
End /* If(xx = 'OK') then Do */
xx = SYSDSN(dsn3)
If(xx = 'OK') then Do
  Nop
End /* If(xx = 'OK') then Do */
Else Do
  iext = 1
End /* If(xx = 'OK') then Do */
If (iext = 1) then Do
  norp = 'NEW'
End /* If (iext = 1) then Do */
If(norp = 'NEW') then Do
  gvar. = ' '
  xx = outtrap('gvar.')
  "DELETE '" ||dsn2 ||"'"
  "DELETE '" ||dsn3 ||"'"
  xx = outtrap(OFF)
  "ATTR OUT LRECL(8Ø) BLKSIZE(64ØØ) RECFM(F B) DSORG(PS)"
  "ALLOC FI(WRITEO) DA('"||dsn2||"') " ,
  "UNIT(SYSDA) USING(OUT) SPACE(3,1) TRACKS" ,
  "CATALOG"
  "ALLOC FI(WRITE3) DA('"||dsn3||"') " ,
  "UNIT(SYSDA) USING(OUT) SPACE(3,1) CYLINDERS" ,
  "CATALOG"
  "FREE ATTRLIST(OUT)"
  "FREE ATTRLIST(WRITEO)"
  "FREE ATTRLIST(WRITE3)"
  "ALLOC FI(OUTP) DA('"dsn2"') MOD"
  "ALLOC FI(OUT3) DA('"dsn3"') MOD"
End /* If(norp = 'NEW') then Do */
Else Do
  gvar. = ' '
  xx = outtrap('gvar.')
  "FREE FI(OUTP)"
  xx = outtrap(OFF)
  "ALLOC FI(OUTP) DA('"dsn2"') MOD"
  "ALLOC FI(OUT3) DA('"dsn3"') MOD"
  "EXECIO * DISKR OUTP (FINIS    STEM ALREAD."
  "EXECIO" alread.Ø "DISKW OUTP (         STEM ALREAD."
End /* If(norp = 'NEW') then Do */
```

```
xt = 0
xt = xt + 1
ttl.xt = "Finish Date/time:" DATE() "-" TIME()
Do jk9 = 1 to xt
  say ttl.jk9
End /* Do jk9 = 1 to xt */
"EXECIO" xh     "DISKW OUTP (         STEM HED."
"EXECIO" nodsp  "DISKW OUTP (         STEM OUTLIN."
If(iali > 0) then Do
  "EXECIO" xf     "DISKW OUTP (         STEM XFH."
End /* If(iali > 0) then Do */
"EXECIO" tal.0  "DISKW OUTP (         STEM TAL."
"EXECIO" xt     "DISKW OUTP (FINIS    STEM TTL."
"EXECIO" xu     "DISKW OUT3 (FINIS    STEM OUTLU."
"FREE FI(OUTP)"
"FREE FI(OUT3)"
exit
/*<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>*/
format_text: procedure  expose indent linp.
mb = 0
Do jk = 1 to linp.0
  Do until linp.jk = " "
    mb = mb + 1
    parse var linp.jk wd.mb linp.jk
    wd.mb = strip(wd.mb,B)
  End /* Do until linp.jk = " " */
End /* Do jk = 1 to linp.0 */

lenl = 78 - indent
xp = 1
x1 = 0
jk1 = 0
wrt.1 = ''
Do until jk1 >= mb
  jk1 = jk1 + 1
  x1 = x1 + length(wd.jk1) + 1
  If(x1 >= (lenl-0) ) then Do
    xp = xp + 1
    wrt.xp = wd.jk1
    x1 = length(wd.jk1)
  End /* If(x1 > lenl) then Do */
  Else Do
    wrt.xp = wrt.xp || ' ' || wd.jk1
  End /* If(x1 > lenl) then Do */
End /*  Do until jk1 > mb */
Do jk = 1 to (xp -1)
  ttt.jk = justify(wrt.jk,lenl)
   say copies(' ',indent) || ttt.jk
End /* Do jk = 1 to xp */
say copies(' ',indent) || wrt.xp
return
```


Automated control card date/time information generation

INTRODUCTION

The following is a simple REXX procedure and JCL to automate the generation of date and time information into control cards – in this case for the LOGREC reporting utility IFCEREP1. The utility came about because our IBM CE had asked us for the ability to run an EREP event report on demand to report back over the past few hours of available LOGREC data. He wanted to have this information when called in for a hardware problem. I decided to write this short REXX EXEC to generate the control cards, allowing him to specify how many hours back he would like the EREP report to start. Since the EXEC was originally written for the IFCEREP1 utility, which does not support the specification of a 4-digit year, the EXEC is not year 2000 compliant. With a little further modfication, it can be made year 2000 compliant, in order to support the generation of control cards for other utilities that do support the specification of 4-digit years.

The basic premise of the EXEC is to accept a parameter specifying how many hours back you want the time to be (in the variable back_time). It checks for a maximum number of hours that can be specified (in the variable max_back) to limit the volume of output generated by the IFCEREP1 utility. The way the EXEC is currently written does not allow for the specification of a back-time of more than 24 hours. It will actually work in the EXEC, but will not produce the desired EREP control cards. The control cards are generated, based on the fact that the EREP utility allows spanning of days by the specification of a start time less than the end time. Users should read the IBM *EREP Reference Manual* for a complete description of the DATE and TIME control cards to fully understand how the EREP program uses them.

The accompanying JCL is meant to be used as a started task so that the specification of how many hours back to report can be specified on the start command. For example, to get a report on the last four hours of EREP information you would issue the MVS start command:

```
S IFCEREPE,HOURS=4
```

The job is a two-step process. The first step runs the TSO Terminal Monitor Program (TMP) in batch to execute the REXX EXEC, which builds the control cards and writes them to the DDNAME of SYSUT2. It accepts the specification of how many hours back the report should go. The second step actually runs the EREP utility using the control cards generated against what should be some EREP history file your installation maintains, pointed to by the ACCIN DD statement.

EREPEVNT

```
/* ————————————  REXX procedure  ————————————— */
parse upper arg back_time           /* number of hours to back up to */
max_back = 8                        /* maximum back-up hours allowed */
if length(back_time) = 0 then do
   say "A valid back time was not specified"
   exit 16
   end
x = verify(back_time,'0123456789')
if x ¬= 0 then do
   say back_time "is not valid 1 or 2 digit number"
   exit 16
   end
if back_time > max_back then do
   say back_time "is greater than maximum allowed time of" max_back
   exit 16
   end
curr_date = date('j')
curr_time = time()
year = substr(curr_date,1,2)
day  = substr(curr_date,3,3)

from_min  = substr(curr_time,4,2)
from_hour = substr(curr_time,1,2)
curr_time = from_hour||from_min

if (from_hour - back_time) < 0 then do
   from_hour = 24 + from_hour - back_time
   from_day  = day - 1
   if from_day < 1 then do
      from_day = 365 + from_day
      year = year - 1
      end
   end
else do
     from_hour = from_hour - back_time
     from_day = day
     end

from_date = year||right(from_day+1000,3)
```

```
from_time = right(from_hour+100,2)||from_min

c.1 = "HIST=Y,ZERO=N,ACC=N,TABSIZE=999K,EVENT=Y"
c.2 = "DATE=("from_date"-"curr_date")"
c.3 = "TIME=("from_time"-"curr_time")"
c.0 = 3
"Execio * diskw SYSUT2 (finis stem c."
exit
```

Sample started task procedure follows:

```
//LOGRECE  PROC HOURS=
//CNTLCARD EXEC PGM=IKJEFT01,REGION=6M,PARM='%EREPEVNT &HOURS'
//SYSTSPRT DD SYSOUT=*,DCB=BLKSIZE=133
//SYSPROC  DD DISP=SHR,DSN=SYSTEMS.EXEC
//SYSUT2   DD DISP=(,PASS),SPACE=(TRK,15),UNIT=VIO,DSN=&&EREPCARD,
//            DCB=(RECFM=F,LRECL=80,BLKSIZE=80)
//SYSTSIN  DD DUMMY,DCB=BLKSIZE=80
//EVENT    EXEC PGM=IFCEREP1,PARM=CARD
//SYSABEND DD SYSOUT=*
//ACCIN    DD DISP=SHR,DSN=P.LOGREC.HISTORY
//DIRECTWK DD UNIT=SYSDA,SPACE=(CYL,(99,99),RLSE)
//EREPPT   DD SYSOUT=a,DCB=(BLKSIZE=1330)
//TOURIST  DD SYSOUT=*,DCB=(BLKSIZE=1330)
//SYSIN    DD DISP=(OLD,DELETE),DSN=&&EREPCARD
```


Generic DASD dataset read and write routines

INTRODUCTION

There are times when it may be useful, or even necessary, to be able to read, and perhaps copy, a DASD dataset of arbitrary format. Whilst standard tools, such as DFSMSdss do exist to help in this area, situations can arise when a more basic approach is appropriate. The two routines presented here, DAREAD and DAWRITE, can be used to read and, if required, make an image copy of any dataset that can be OPENed with a standard OPEN macro (ie anything other than VSAM), and read and written using EXCP.

DAREAD

The DAREAD routine builds a channel program to read a single physical record at a time and executes it via the EXCP macro. It is not influenced by anything other than what it finds on the track it is reading – for example it takes no account of the ‘official end-of-data' indicator in the dataset's format-1 DSCB DS1LSTAR field, nor does it treat end-of-file records as anything other than what they really are, (records with zero data length). DAREAD will read a PDS from physical start to physical end, ie the directory blocks (including the hardware keys), the EOF following the directory, and all the individual member records, whether live or deleted, and all the embedded EOFs. DAREAD will also read records larger than the 32KB blocksize limit imposed by standard access methods (eg QSAM).

DAWRITE

The DAWRITE routine builds a channel program to write a single physical record, with or without hardware keys, and executes it via the EXCP macro. The length of each record, and its key, if any, are specified on a record-by-record basis and may take any values legal for the type of DASD being written to. In principle it could be used to write full-track blocked datasets because, like DAREAD, it is not subject to the 32KB blocksize limit of conventional access methods. (If written, such a dataset could then be read back by DAREAD, but probably nothing else.)

EOF records are written simply by specifying a record with both key length and data length set to zero. Thus DAWRITE can be used to physically duplicate an entire PDS, directory included, by writing image copies of the records read by DAREAD.

SUGGESTED USAGE

DAREAD and DAWRITE can be called from any programming language that supports standard OS/370 linkage conventions, and have no addressing or residency mode restrictions. An outline of a typical program using these routines is as follows:

1 Allocate the input dataset by an appropriate method – JCL or via dynamic allocation.

2 Allocate the output dataset similarly. Depending on what the purpose of the program is, the DCB parameters may or may not be the same as those of the input dataset.

3 Call DAREAD to read the next record. Note that the convention used is that the first track of the dataset is track 1 and the first record on a track is record 1 (DAREAD does not touch the standard record 0 that is written on every track when the DASD is formatted).

4 If the record is read successfully, write it to the output dataset in an appropriate manner. For example, if making an image copy of a dataset, use DAWRITE; if simply reading the dataset to see what it contains, use QSAM for example to write it or some subset of it. Increment the record counter and loop back to step 3. to read the next record from the current track.

5 If the record was not found, increment the track counter, reset the record counter to 1, and loop back to step 3.

6 If DAREAD indicates that the track specified is beyond the last allocated track, this indicates the physical end of the dataset. If using DAWRITE to make a copy, call it to complete and close the output DCB.

Such a program has several uses. It could be used to read a corrupted dataset (eg a PDS with a damaged directory) that is unreadable by normal access methods such as PSAM or QSAM, and write a new copy, possibly having repaired the damage. By writing a copy via QSAM to a RECFM=U, BLKSIZE=32760 dataset or to JES SYSOUT, it is simple to obtain a BROWSEable copy. This can be very useful when trying to diagnose problems.

As indicated above, such a program can be used to make an image copy of any file, including a PDS. To do this, the output file should be allocated with the same size and DCB parameters as the input file; in the case of a PDS this includes the number of directory blocks. Then run the program, with DAWRITE writing an image of every record, including ends-of file, read from the input dataset by DAREAD. In the case of a PDS, this will overlay the empty directory blocks created when the output file was allocated with those read from the input dataset, and then copy in all the member records.

A DAREAD/DAWRITE program can also be used to recover the contents of a deleted dataset, provided that it is known which tracks on which device it occupied, and that those tracks have not been overwritten. In such a case, an empty *sequential* dataset (*not* a PDS or the first part of the dataset will be overlaid with empty directory blocks) should be allocated over the relevant tracks, and the output dataset should be allocated with the specifications of the original dataset. The DAREAD/DAWRITE program should then be run and will copy all the data from the original location to the new dataset, so recovering the data.

OPERATIONAL ENVIRONMENT

DAREAD and DAWRITE have no special authorization requirements and can be called from any problem-state program. All that is necessary to make them available is to assemble and link-edited them into a suitable load library. It would, however, be advisable to restrict access to these routines to those people who need them, such as storage administrators; they are not really intended for general use.

The versions of DAREAD and DAWRITE presented here should run under any version of MVS/XA or MVS/ESA, and DFP or DFSMS/MVS; they are known to work on MVS/XA 2.2.3 and DFP 2.4, MVS/ESA 4.2.2 and DFP 3.3, and MVS/ESA 5.1.0 and DFSMS/MVS 1.2.0.

DAREAD

```
         TITLE 'DAREAD - Generic DASD Dataset Read Routine'
*********************************************************************
*  Subroutine DAREAD
*  ————————-
*  This routine will read the specified record from  a  direct  access
*  dataset of any type.  The record  is  specified  by its location in
*  terms of which track it is located on, and its relative position on
*  that track.   The first track of the dataset is track 1;  the first
*  data record on a track is record 1.
*
*  This routine is not sensitive to EOF markers  and,   called  in the
*  appropriate manner, will read the entire allocated extent(s) of any
*  dataset.  For example, it will read a PDS from end to end,  includ-
*  ing the directory blocks  and  all  deleted  members.  It will also
*  read an 'empty' dataset - eg a dataset allocated but never written.
*
*  This routine is designed  for  robustness  rather than efficiency -
```

```
*  it is intended more as a diagnostic or recovery tool.  For  example
*  it can be used to copy  a  damaged  or otherwise unreadable dataset
*  to a BROWSEable copy.   In this context,   the copy dataset is best
*  allocated with RECFM=U and BLKSIZE=32760.  Once  this readable copy
*  has been made,  recovery actions specific to the nature of the data
*  can be taken, using normal access methods.
*
*  This routine may  be called from any high-level language that  uses
*  standard OS/370 linkage conventions.  It  has no addressing or res-
*  idency mode restrictions.
*
*  ARGUMENTS : CALL DAREAD(DDNAME,ITRACK,IREC,NBYTES,RECORD,IOSTAT)
*  ----   ----------------------------
*
*    DDNAME  :  DDNAME of pre-allocated dataset               ( INPUT )
*    ITRACK  :  Track number (first track is track 1)         ( INPUT )
*      IREC  :  Record number on track                        ( INPUT )
*    NBYTES  :  Number of bytes read (if IOSTAT=0) -         ( OUTPUT )
*               (1) : Key length (zero if record not keyed)
*               (2) : Data length
*    RECORD  :  Record (Key + Data)                          ( OUTPUT )
*    IOSTAT  :  Error flag                                   ( OUTPUT )
*
*  The value of IOSTAT should always be checked on return, and should
*  be used to control the incrementing of ITRACK and IREC. The values
*  IOSTAT may have are :
*
*     IOSTAT =  0 : Record read successfully; length is in NBYTES.
*               4 : Dataset open failed (probably not allocated).
*               8 : EOF (zero length record) - move to next record.
*              12 : Record not found - move to next track.
*              16 : ITRACK is outside the allocated extent(s) of the
*                   dataset - stop reading.
*              20 : Real I/O error (should never happen).
*
*  Note that NBYTES is only set if IOSTAT is zero,  and should not be
*  used unless that is the case. Providing that it is set, the RECORD
*  area will contain NBYTES(1) bytes of key data,   followed  contig-
*  uously by NBYTES(2) bytes of data.
*
*  Operational requirements:
*
*    STATE    : Problem
*    KEY      : 8
*    APF      : No
*    AMODE    : 31
*    RMODE    : Any
*    LOCATION : Callable subroutine
*********************************************************************
*
         EJECT
DAREAD   CSECT
```

```
DAREAD   AMODE 31
DAREAD   RMODE ANY
*
RØ       EQU   Ø                          * WORK REGISTER
R1       EQU   1                          * @(ARGUMENT LIST)
R2       EQU   2                          * @(DDNAME)
R3       EQU   3                          * @(ITRACK)
R4       EQU   4                          * @(IREC) )
R5       EQU   5                          * @(NBYTES)
R6       EQU   6                          * @(BUFFER)
R7       EQU   7                          * @(IOSTAT)
R8       EQU   8                          * WORK REGISTER
R9       EQU   9                          * WORK REGISTER
R1Ø      EQU   1Ø                         * WORK REGISTER
R11      EQU   11                         * WORK REGISTER
R12      EQU   12                         * BASE REGISTER
R13      EQU   13                         * OUR SAVEAREA
R14      EQU   14                         * RETURN ADDRESS
R15      EQU   15                         * ENTRY ADDRESS/RETURN CODE
*
         USING *,R15                      * ADDRESSABILITY
         B     START                      * BRANCH TO START OF CODE
         DC    AL1(LASTL-FIRSTL)          * LENGTH OF HEADER TEXT
FIRSTL   EQU   *
         DC    CL8'DAREAD  '
LASTL    EQU   *
         DC    C' '
         DC    CL8'&SYSDATE'
         DC    C' '
         DC    CL5'&SYSTIME'
         DROP  R15                        * FINISHED WITH R15
         DS    ØF                         * ALIGN TO FULL WORD BOUNDARY
*
*********************************************************************
* ADDRESSABILITY AND LINKAGE
*********************************************************************
*
START    EQU   *
         STM   R14,R12,12(R13)            * SAVE REGISTERS IN HSA
         LR    R12,R15                    * LOAD BASE REGISTER
         USING DAREAD,R12                 * AND DEFINE ADDRESSIBILITY
*
         LR    R11,R13                    * R11 = ADDRESS OF HSA
         LA    R13,SAVEAREA               * R13 = ADDRESS OF LSA
         ST    R11,4(R13)                 * STORE HSA ADDRESS
         ST    R13,8(R11)                 * STORE LSA ADDRESS
*
* GET ARGUMENT ADDRESSES
*
         LM    R2,R7,Ø(R1)                * LOAD ARG ADDRESSES
*
* IS THIS THE FIRST CALL ?
```

```
*
         ICM   R1,B'1111',ADEB           * FIRST CALL ?
         BNZ   DOREAD                    * NO
         EJECT
*
*********************************************************************
* ON THE FIRST CALL, PERFORM SOME ONE-OFF INITIALIZATION PROCESSING
*********************************************************************
*
* GET @(SECTOR-CONVERT RTN) AND @(REAL-BLOCK-ADDRESS RTN) FROM CVT
*
         L     R11,16                    * R11 = ADDRESS OF CVT
         USING CVT,R11                   * DEFINE CVT ADDRESSABILITY
*
         MVC   AØSCR1,CVTØSCR1           * @(SECTOR-CONVERT ROUTINE)
         MVC   APCNVT,CVTPCNVT           * @(REAL-BLOCK-ADDRESS RTNE)
         DROP  R11                       * FINISHED WITH CVT
*
* IF NEED BE GET BELOW-LINE STORAGE FOR DCB, IOB, AND CHANNEL PROGRAM
*
         TM    ADCB,X'FF'                * ARE WE ABOVE THE LINE ?
         BZ    BUILDCBS                  * IF NOT NO NEED TO MOVE DCB
*
         GETMAIN RU,LV=LDCB,BNDRY=DBLWD,LOC=(BELOW,ANY)
*
         ST    R1,ADCB                   * SAVE BELOW-LINE DCB ADDRESS
         MVC   Ø(LDCB,R1),DCB            * MOVE DCB BELOW LINE
*
         GETMAIN RU,LV=LIOB,BNDRY=DBLWD,LOC=(BELOW,ANY)
*
         ST    R1,AIOB                   * SAVE IOB ADDRESS
         MVC   Ø(LIOB,R1),IOBAREA        * MOVE IOB BELOW LINE
*
         GETMAIN RU,LV=LCHPROG,BNDRY=DBLWD,LOC=(BELOW,ANY)
*
         ST    R1,ACHPROG                * SAVE @(CHANNEL PROGRAM)
         MVC   Ø(LCHPROG,R1),CHPROG      * MOVE CHPROG BELOW LINE
*
*-------------------------------------
* COMPLETE DCB, IOB, AND CHANNEL PROGRAM
*-------------------------------------
*
BUILDCBS EQU   *
         L     R11,ADCB                  * R11 = DCB ADDRESS
         USING IHADCB,R11                * DEFINE DCB ADDRESSABILITY
*
         MVC   DCBDDNAM,Ø(R2)            * MOVE DDNAME INTO DCB
*
         L     R1Ø,AIOB                  * R1Ø = IOB ADDRESS
         USING IOB,R1Ø                   * IOB ADDRESSABILITY
*
         MVI   IOBFLAG1,X'C2'            * CMND+DATA CHAINING,UNRELATED
```

```
*
         STCM  R11,B'Ø111',IOBDCBPT      * INSERT @(DCB) INTO IOB
*
         L     R9,ACHPROG                * @(CHANNEL PROGRAM)
         STCM  R9,B'Ø111',IOBSTART       * INSERT @(CH PROG) INTO IOB
*
         LA    R8,IOBECB                 * @(IOBECB)
         STCM  R8,B'Ø111',IOBECBPT       * INSERT @(IOBECB) INTO IOB
*
         LA    R8,6Ø(R9)                 * INSERT @(SECTOR NUMBER) ...
         STCM  R8,B'Ø111',1(R9)          * ... INTO SET SECTOR CCW
*
         LA    R8,IOBSEEK+3              * INSERT @(SEEK ADDRESS) ...
         STCM  R8,B'Ø111',9(R9)          * ... INTO SEARCH ID EQUAL CCW
*
         LA    R8,8(R9)                  * INSERT @(SEARCH ID CCW) ...
         STCM  R8,B'Ø111',17(R9)          * ... INTO TIC CCW
*
         LA    R8,48(R9)                 * INSERT @(COUNT) ...
         STCM  R8,B'Ø111',25(R9)         * ... INTO RCKD1 CCW
*
         LA    R8,56(R9)                 * INSERT @(IDAW) ...
         STCM  R8,B'Ø111',33(R9)         * ... INTO RCKD2 CCW
*
         LA    R8,6Ø(R9)                 * INSERT @(SECTOR NUMBER) ...
         STCM  R8,B'Ø111',41(R9)         * ... INTO READ SECTOR CCW
*
         DROP  R1Ø                       * FINISHED WITH IOB
*
*-------------------------------------
* OPEN THE DATASET
*-------------------------------------
*
         OPEN  ((R11),INPUT),MODE=31     * OPEN DATASET FOR INPUT
*
         TM    DCBOFLGS,DCBBIT3          * BIT 3 SHOULD BE 1
         BZ    ERROR4                    * ITS NOT SO AN ERROR OCCURRED
*
* GET DEB ADDRESS FROM DCB AND SAVE IT
*
         SR    R1,R1                     * R1 = ...
         ICM   R1,B'Ø111',DCBDEBA        * ... DEB ADDRESS
         ST    R1,ADEB                   * SAVE IT
*
         DROP  R11                       * FINISHED WITH DCB
         EJECT
*
*********************************************************************
* READ DATA
*********************************************************************
*
DOREAD   EQU   *
```

```
         L     R3,Ø(R3)                  * GET TRACK NUMBER
         BCTR  R3,Ø                      * FIRST TRACK IS TRACK Ø
         STH   R3,TTRZ                   * INSERT TT INTO TTRZ
*
         L     R4,Ø(R4)                  * GET RECORD NUMBER ON TRACK
         BCTR  R4,Ø                      * SUBTRACT 1
         STC   R4,TTRZ+2                 * INSERT R INTO TTRZ
*
         L     R9,ACHPROG                * INSERT BUFFER ADDRESS ...
         ST    R6,56(,R9)                * ... INTO IDAW
*
         LTR   R4,R4                     * IF IREC > 1 ...
         BP    MBBCCHHR                  * ... LEAVE PREVIOUS SECNUM
*
         MVI   6Ø(R9),X'ØØ'              * NEW TRACK : RESET SECNUM
*
* COMPUTE ACTUAL DEVICE ADDRESS (MBBCCHHR) OF RECORD
*
MBBCCHHR EQU   *
         L     R1Ø,AIOB                  * R1Ø = IOB ADDRESS
         USING IOB,R1Ø                   * IOB ADDRESSABILITY
*
         STM   R9,R13,SAVE913            * SAVE R9 - R13
         L     RØ,TTRZ                   * RØ = TTRZ OF REQUIRED BLOCK
         L     R1,ADEB                   * R1 = DEB ADDRESS
         LA    R2,IOBSEEK                * R2 = @(MBBCCHHR)
*
         LR    R8,R12                    * SAVE BASE REGISTER
         L     R15,APCNVT                * @(REAL-DEVICE-ADDRESS RTN)
         BASR  R14,R15                   * BRANCH TO IT
*
         LR    R12,R8                    * RESTORE BASE REGISTER
         LM    R9,R13,SAVE913            * AND RESTORE R9 - R13
         LTR   R15,R15                   * TEST RETURN CODE
         BNZ   EOD                       * NOT ZERO MEANS ERROR
*
* READ THE RECORD
*
         XC    IOBECB,IOBECB             * CLEAR ECB
*
         EXCP  (R1Ø)                     * EXECUTE CHANNEL PROGRAM
*
         WAIT  ECB=IOBECB                * WAIT ON IOBECB
*
         TM    IOBECB,X'7F'              * TEST FOR SUCCESSFULL READ
         BO    READOK                    * MATCH MEANS READ OK
*
*-------------------------------------
* I/O ERROR DETECTED - ANALYSE THE SITUATION:
*-------------------------------------
*
```

```
* UNIT CHECK + NO-RECORD-FOUND MEANS RECORD DOES NOT EXIST
*
         TM    IOBCSW+3,X'Ø2'            * UNIT CHECK ?
         BNO   TRYEOF                    * NO, SO TRY FOR REAL EOF
*
         TM    IOBSENS1,X'Ø8'            * 'NO RECORD FOUND' ?
         BO    EOT                       * YES - END OF DATA ON TRACK
         B     ERROR2Ø                   * ANYTHING ELSE IS FATAL ERROR
*
* A RECORD WITH ZERO DATA LENGTH IS AN END-OF-FILE
*
TRYEOF   EQU   *
         SR    R9,R9                     * R9 = ...
         ICM   R9,B'ØØ11',RCKD2+6        * ... CCW COUNT
         SR    R8,R8                     * R8 = ...
         ICM   R8,B'ØØ11',IOBCSW+5       * ... RESIDUAL COUNT
*
         CR    R9,R8                     * COMPARE CCW & RESIDUAL COUNT
         BE    EOF                       * EQUAL MEANS EOF RECORD
         B     ERROR2Ø                   * ANYTHING ELSE IS FATAL ERROR
*
*-------------------------------------
* RECORD READ OK. RETURN KEY AND DATA LENGTHS TO CALLER
*-------------------------------------
*
READOK   EQU   *
         L     R9,ACHPROG                * @(CHANNEL PROGRAM)
*
         SR    RØ,RØ                     * RØ = ...
         IC    RØ,52(,R9)                * ... R ACTUALLY READ
         SR    R1,R1                     * R1 = ...
         IC    R1,IOBSEEK+7              * ... R USED IN SEARCH ID
         SR    RØ,R1                     * IF R(READ) ¬= ...
         C     RØ,F1                     * ... R(SEARCH) + 1 ...
         BNE   EOT                       *     ... WE HAVE PASSED EOT
*
         SR    R8,R8                     * COPY ...
         IC    R8,53(,R9)                * ... KEY LENGTH ...
         ST    R8,Ø(,R5)                 *     ... TO CALLER
         ICM   R8,B'ØØ11',54(R9)         * COPY DATA LENGTH ...
         ST    R8,4(,R5)                 * ... TO CALLER
*
         SR    R15,R15                   * RC = Ø
*
         DROP  R1Ø                       * FINISHED WITH IOB
         EJECT
*
*********************************************************************
* RETURN TO CALLER
*********************************************************************
*
RETURN   EQU   *
```

```
         ST    R15,Ø(R7)                * STORE IOSTAT FOR CALLER
         L     R13,4(R13)               * RESTORE ADDRESS OF HSA
         L     R14,12(R13)              * RESTORE R14
         LM    RØ,R12,2Ø(R13)           * RESTORE RØ-R12
         BR    R14                      * AND RETURN
         EJECT
*
*********************************************************************
* ERROR CONDITIONS
*********************************************************************
*
ERROR4   EQU   *
         LA    R15,4                    * OPEN FAILURE
         B     RETURN                   * RETURN
*
EOF      EQU   *
         LA    R15,8                    * EOF
         B     RETURN                   * RETURN
*
EOT      EQU   *
         LA    R15,12                   * NO RECORD FOUND
         B     RETURN                   * RETURN
*
EOD      EQU   *
         LA    R15,16                   * EXTENT VIOLATION
         B     RETURN                   * RETURN
*
ERROR2Ø  EQU   *
         LA    R15,2Ø                   * I/O ERROR
         B     RETURN                   * RETURN
         EJECT
*
*********************************************************************
* CONSTANTS, VARIABLES AND DATA AREAS
*********************************************************************
*
         DS    ØD
         DC    CL8'SAVEAREA'
SAVEAREA DS    18F
SAVE913  DS    5F
*
F1       DC    F'1'
AØSCR1   DS    A                        * @(SECTOR CONVERT ROUTINE)
APCNVT   DS    A                        * @(REAL BLOCK ADDRESS RTN)
TTRZ     DC    XL4'ØØØØØØØØ'            * TTRZ OF REQUESTED RECORD
*
ADCB     DC    A(DCB)
ADEB     DC    A(Ø)
AIOB     DC    A(IOBAREA)
ACHPROG  DC    A(CHPROG)
*
```

```
         DS    ØF
DCB      DCB   DDNAME=DUMMY,DSORG=PS,MACRF=E
LDCB     EQU   *-DCB
*
         DS    ØF
IOBAREA  DC    (LIOB)X'ØØ'
*
         DS    ØD
CHPROG   EQU   *
SETSECT  CCW   X'23',SECNUM,X'4Ø',1      * SET SECTOR
SEARCHID CCW   X'31',Ø,X'4Ø',5           * SEARCH ID EQUAL
TIC      CCW   X'Ø8',SEARCHID,Ø,Ø        * RETRY UNTIL SUCCESSFULL
RCKD1    CCW   X'1E',COUNT,X'8Ø',8       * READ COUNT ...
RCKD2    CCW   X'1E',IDAW,X'64',65535    * ... KEY AND DATA
READSECT CCW   X'22',SECNUM,X'ØØ',1      * READ SECTOR
*
COUNT    DS    D                         * COUNT FIELD
CCHH     EQU   COUNT,4                   *
R        EQU   COUNT+4,1                 *
K        EQU   COUNT+5,1                 *
DD       EQU   COUNT+6,2                 *
IDAW     DS    F                         * INDIRECT DATA ADDRESS WORD
SECNUM   DC    XL1'ØØ'                   * SECTOR NUMBER
LCHPROG  EQU   *-CHPROG
*
IOB      DSECT
IOBFLAG1 DS    XL1                       * CHAINING/UNRELATED BITS
IOBFLAG2 DS    XL1                       * NOT USED HERE
IOBSENSE DS    ØXL2                      * SENSE BYTES
IOBSENSØ DS    XL1                       *   SENSE BYTE 1
IOBSENS1 DS    XL1                       *   SENSE BYTE 2
IOBECBCC DS    XL1                       * FIRST BYTE OF COMP. CODE
IOBECBPT DS    AL3                       * ECB ADDRESS
IOBFLAG3 DS    XL1                       * SYSTEM USE ONLY
IOBCSW   DS    XL7                       * CHANNEL STATUS WORD
IOBSIOCC DS    XL1                       * START SUBCHANNEL DATA
IOBSTART DS    AL3                       * CHANNEL PROGRAM ADDRESS
         DS    XL1                       * RESERVED
IOBDCBPT DS    AL3                       * DCB ADDRESS
IOBRESTR DS    XL1                       * USED FOR ERROR RECOVERY
         DS    XL3                       * USED FOR ERROR RECOVERY
IOBINCAM DS    XL2                       * USED FOR MAG TAPE ONLY
         DS    XL2                       * RESERVED
IOBSEEK  DS    XL8                       * SEEK ADDRESS (MBBCCHHR)
*
IOBECB   DS    F                         * ECB
LIOB     EQU   *-IOB
*
         PRINT NOGEN
         CVT   DSECT=YES                 * CVT MAPPING MACRO
         PRINT NOGEN
```

```
         DCBD  DSORG=PS,DEVD=DA          * DCB MAPPING MACRO
*
         END
```

DAWRITE

```
TITLE 'DAWRITE - Generic DASD Dataset Write Routine'
*********************************************************************
*  Subroutine DAWRITE
*  ─────────
*  This routine will write the specified record  to  a  direct access
*  dataset of any type.  The record  is  specified by its location in
*  terms of which track it is located on and its relative position on
*  that track.  The first track of the dataset is track 1;  the first
*  data record on a track is record 1.
*
*  This routine is designed  for  robustness rather than efficiency -
*  it is intended more as a diagnostic or recovery tool.  For example
*  it can be used to copy  a  damaged or otherwise unreadable dataset
*  to a BROWSEable copy.   In this context,  the copy dataset is best
*  allocated with RECFM=U and BLKSIZE=32760.  Once this readable copy
*  has been made, recovery actions specific to the nature of the data
*  can be taken, using normal access methods.
*
*  This routine may  be called from any high-level language that uses
*  standard OS/370 linkage conventions.  It has no addressing or res-
*  idency mode restrictions.
*
*  ARGUMENTS : CALL DAWRITE(DDNAME,ITRACK,IREC,NBYTES,RECORD,IOSTAT)
*  ─────   ──────────────────────────-
*
*    DDNAME  :  DDNAME of pre-allocated dataset              ( INPUT )
*    ITRACK  :  Track number (first track is track 1)        ( INPUT )
*      IREC  :  Record number on track (first is record 1)   ( INPUT )
*    NBYTES  :  Number of bytes to be written (see note)     ( INPUT )
*               (1) Key length (Set to zero if record not keyed)
*               (2) Data length
*    RECORD  :  Record to be written (Key + Data)            ( INPUT )
*    IOSTAT  :  Error flag                                  ( OUTPUT )
*
*  Note: NBYTES(2) should have one of the following values:
*        1) > 0 : Write data record of this length
*        2)   0 : Write an end-of file
*        3) < 0 : Close the file
*
*  Note: RECORD should contain NBYTES(1) bytes for the key,  followed
*        by NBYTES(2) bytes for the data component.
*
*  The value of IOSTAT should always be checked on return. the values
*  IOSTAT may have are:
*
```

```
*     IOSTAT =  Ø : Record written successfully.
*               4 : Dataset open failed (probably not allocated).
*               8 : Write sequence error - record IREC-1 not found
*              12 : Track overflow.
*              16 : ITRACK is outside the allocated extent(s) of the
*                   dataset - stop writing.
*              2Ø : Real I/O error (should never happen).
*
*  Operational requirements:
*
*    STATE    : Problem
*    KEY      : 8
*    APF      : No
*    AMODE    : 31
*    RMODE    : Any
*    LOCATION : Callable subroutine
*********************************************************************
*
         EJECT
DAWRITE  CSECT
DAWRITE  AMODE 31
DAWRITE  RMODE ANY
*
RØ       EQU   Ø                        * WORK REGISTER
R1       EQU   1                        * @(ARGUMENT LIST)
R2       EQU   2                        * @(DDNAME)
R3       EQU   3                        * @(ITRACK)
R4       EQU   4                        * @(IREC) )
R5       EQU   5                        * @(NBYTES)
R6       EQU   6                        * @(BUFFER)
R7       EQU   7                        * @(IOSTAT)
R8       EQU   8                        * WORK REGISTER
R9       EQU   9                        * WORK REGISTER
R1Ø      EQU   1Ø                       * WORK REGISTER
R11      EQU   11                       * WORK REGISTER
R12      EQU   12                       * BASE REGISTER
R13      EQU   13                       * OUR SAVEAREA
R14      EQU   14                       * RETURN ADDRESS
R15      EQU   15                       * ENTRY ADDRESS/RETURN CODE
*
         USING *,R15                    * ADDRESSABILITY
         B     START                    * BRANCH TO START OF CODE
         DC    AL1(LASTL-FIRSTL)        * LENGTH OF HEADER TEXT
FIRSTL   EQU   *
         DC    CL8'DAWRITE '
LASTL    EQU   *
         DC    C' '
         DC    CL8'&SYSDATE'
         DC    C' '
         DC    CL5'&SYSTIME'
         DROP  R15                      * FINISHED WITH R15
         DS    ØF                       * ALIGN TO FULL WORD BOUNDARY
```

```
*
*********************************************************************
* ADDRESSABILITY AND LINKAGE
*********************************************************************
*
START    EQU   *
         STM   R14,R12,12(R13)          * SAVE REGISTERS IN HSA
         LR    R12,R15                  * LOAD BASE REGISTER
         USING DAWRITE,R12              * AND DEFINE ADDRESSIBILITY
*
         LR    R11,R13                  * R11 = ADDRESS OF HSA
         LA    R13,SAVEAREA             * R13 = ADDRESS OF LSA
         ST    R11,4(R13)               * STORE HSA ADDRESS
         ST    R13,8(R11)               * STORE LSA ADDRESS
*
* GET ARGUMENT ADDRESSES
*
         LM    R2,R7,Ø(R1)              * LOAD ARG ADDRESSES
*
* IS THIS THE FIRST CALL ?
*
         ICM   R1,B'1111',ADEB          * FIRST CALL ?
         BNZ   DOWRITE                  * NO
         EJECT
*
*********************************************************************
* ON THE FIRST CALL, PERFORM SOME ONE-OFF INITIALIZATION PROCESSING
*********************************************************************
*
* GET @(SECTOR-CONVERT RTN) AND @(REAL-BLOCK-ADDRESS RTN) FROM CVT
*
         L     R11,16                   * R11 = ADDRESS OF CVT
         USING CVT,R11                  * DEFINE CVT ADDRESSABILITY
*
         MVC   AØSCR1,CVTØSCR1          * @(SECTOR-CONVERT ROUTINE)
         MVC   APCNVT,CVTPCNVT          * @(REAL-BLOCK-ADDRESS RTNE)
         DROP  R11                      * FINISHED WITH CVT
*
* IF NEED BE GET BELOW-LINE STORAGE FOR DCB, IOB, AND CHANNEL PROGRAM
*
         TM    ADCB,X'FF'               * ARE WE ABOVE THE LINE ?
         BZ    BUILDCBS                 * IF NOT NO NEED TO MOVE DCB
*
         GETMAIN RU,LV=LDCB,BNDRY=DBLWD,LOC=(BELOW,ANY)
*
         ST    R1,ADCB                  * SAVE BELOW-LINE DCB ADDRESS
         MVC   Ø(LDCB,R1),DCB           * MOVE DCB BELOW LINE
*
         GETMAIN RU,LV=LIOB,BNDRY=DBLWD,LOC=(BELOW,ANY)
*
         ST    R1,AIOB                  * SAVE IOB ADDRESS
         MVC   Ø(LIOB,R1),IOBAREA       * MOVE IOB BELOW LINE
```

```
*
         GETMAIN RU,LV=LCHPROG,BNDRY=DBLWD,LOC=(BELOW,ANY)
*
         ST    R1,ACHPROG                * SAVE @(CHANNEL PROGRAM)
         MVC   Ø(LCHPROG,R1),CHPROG      * MOVE CHPROG BELOW LINE
*
*-------------------------------------
* COMPLETE DCB, IOB, AND CHANNEL PROGRAM
*-------------------------------------
*
BUILDCBS EQU   *
         L     R11,ADCB                  * R11 = DCB ADDRESS
         USING IHADCB,R11                * DEFINE DCB ADDRESSABILITY
*
         MVC   DCBDDNAM,Ø(R2)            * MOVE DDNAME INTO DCB
*
         L     RIØ,AIOB                  * RIØ = IOB ADDRESS
         USING IOB,RIØ                   * IOB ADDRESSABILITY
*
         STCM  R11,B'Ø111',IOBDCBPT      * INSERT @(DCB) INTO IOB
*
         L     R9,ACHPROG                * @(CHANNEL PROGRAM)
         STCM  R9,B'Ø111',IOBSTART       * INSERT @(CH PROG) INTO IOB
*
         LA    R8,IOBECB                 * @(IOBECB)
         STCM  R8,B'Ø111',IOBECBPT       * INSERT @(IOBECB) INTO IOB
*
         LA    R8,6Ø(R9)                 * INSERT @(SECTOR NUMBER) ...
         STCM  R8,B'Ø111',1(R9)          * ... INTO SET SECTOR CCW
*
         LA    R8,IOBSEEK+3              * INSERT @(SEEK ADDRESS) ...
         STCM  R8,B'Ø111',9(R9)          * ... INTO SEARCH ID EQUAL CCW
*
         LA    R8,8(R9)                  * INSERT @(SEARCH ID CCW) ...
         STCM  R8,B'Ø111',17(R9)          * ... INTO TIC CCW
*
         LA    R8,48(R9)                 * INSERT @(COUNT) ...
         STCM  R8,B'Ø111',25(R9)         * ... INTO WCKD1 CCW
*
         LA    R8,56(R9)                 * INSERT @(IDAW) ...
         STCM  R8,B'Ø111',33(R9)         * ... INTO WCKD2 CCW
*
         LA    R8,6Ø(R9)                 * INSERT @(SECTOR NUMBER) ...
         STCM  R8,B'Ø111',41(R9)         * ... INTO READ SECTOR CCW
*
         DROP  RIØ                       * FINISHED WITH IOB
*
*-------------------------------------
* OPEN THE DATASET
*-------------------------------------
*
         OPEN  ((R11),OUTPUT),MODE=31    * OPEN DATASET FOR OUTPUT
```

```
*
         TM    DCBOFLGS,DCBBIT3          * BIT 3 SHOULD BE 1
         BZ    ERROR4                    * ITS NOT SO AN ERROR OCCURRED
*
* GET DEB ADDRESS FROM DCB AND SAVE IT
*
         SR    R1,R1                     * R1 = ...
         ICM   R1,B'Ø111',DCBDEBA        * ... DEB ADDRESS
         ST    R1,ADEB                   * SAVE IT
*
         DROP  R11                       * FINISHED WITH DCB
         EJECT
*
*********************************************************************
* WRITE DATA
*********************************************************************
*
DOWRITE  EQU   *
         ICM   R1,B'1111',4(R5)          * GET NBYTES(2)
         BM    CLOSE                     * IF -VE CLOSE DATASET
*
         L     R3,Ø(R3)                  * GET TRACK NUMBER
         BCTR  R3,Ø                      * FIRST TRACK IS TRACK Ø
         STH   R3,TTRZ                   * INSERT TT INTO TTRZ
*
         L     R4,Ø(R4)                  * GET RECORD NUMBER ON TRACK
         ST    R4,IREC                   * SAVE IT
         BCTR  R4,Ø                      * SUBTRACT 1
         STC   R4,TTRZ+2                 * INSERT R INTO TTRZ
*
         L     R9,ACHPROG                * INSERT BUFFER ADDRESS ...
         ST    R6,56(R9)                 * ... INTO IDAW
*
         LTR   R4,R4                     * IF IREC > 1 ...
         BP    MBBCCHHR                  * ... LEAVE PREVIOUS SECNUM
*
         MVI   6Ø(R9),X'ØØ'              * NEW TRACK : RESET SECNUM
*
* COMPUTE ACTUAL DEVICE ADDRESS (MBBCCHHR) OF RECORD
*
MBBCCHHR EQU   *
         L     R1Ø,AIOB                  * R1Ø = IOB ADDRESS
         USING IOB,R1Ø                   * IOB ADDRESSABILITY
*
         STM   R9,R13,SAVE913            * SAVE R9 - R13
         L     RØ,TTRZ                   * RØ = TTRZ OF REQUIRED BLOCK
         L     R1,ADEB                   * R1 = DEB ADDRESS
         LA    R2,IOBSEEK                * R2 = @(MBBCCHHR)
*
         LR    R8,R12                    * SAVE BASE REGISTER
         L     R15,APCNVT                * @(REAL-DEVICE-ADDRESS RTN)
         BASR  R14,R15                   * BRANCH TO IT
*
```

```
         LR    R12,R8                    * RESTORE BASE REGISTER
         LM    R9,R13,SAVE913            * AND RESTORE R9 - R13
         LTR   R15,R15                   * TEST RETURN CODE
         BNZ   EOD                       * NOT ZERO MEANS ERROR
*
* BUILD THE COUNT FIELD (X'CCHHRKDD') AND TAKE CARE OF EOF REQUEST
*
         MVC   48(4,R9),IOBSEEK+3        * MOVE IN CCHH PART OF COUNT
         MVC   52(1,R9),IREC+3           * MOVE IN R    PART OF COUNT
         MVC   53(1,R9),3(R5)            * MOVE IN K    PART OF COUNT
         MVC   54(2,R9),6(R5)            * MOVE IN DD   PART OF COUNT
*
         ICM   R1,B'1111',4(R5)          * IS NBYTES(2) = Ø ?
         BNZ   DATACCW                   * NO
*
         MVI   IOBFLAG1,X'42'            * CMND CHAINING,UNRELATED
         MVI   28(R9),X'ØØ'              * CLEAR WCKD1 CCW FLAGS
         B     EXCP                      * AND JUMP
*
DATACCW  EQU   *
         MVI   IOBFLAG1,X'C2'            * CMND+DATA CHAINING,UNRELATED
         MVI   28(R9),X'8Ø'              * SWITCH ON DATA CHAINING
         L     R8,Ø(R5)                  * TOTAL CCW COUNT ...
         A     R8,4(R5)                  * ... = L'KEY + L'DATA
         STH   R8,38(,R9)                * UPDATE COUNT IN WCKD2 CCW
*
* WRITE THE RECORD
*
EXCP     EQU   *
         XC    IOBECB,IOBECB             * CLEAR ECB
*
         EXCP  (R1Ø)                     * EXECUTE CHANNEL PROGRAM
*
         WAIT  ECB=IOBECB                * WAIT ON IOBECB
*
         TM    IOBECB,X'7F'              * TEST FOR SUCCESSFULL WRITE
         BO    WRITEOK                   * MATCH MEANS WRITE OK
*
*-------------------------------------
* I/O ERROR DETECTED - ANALYSE THE SITUATION :
*-------------------------------------
*
* UNIT CHECK + NO-RECORD-FOUND MEANS PRIOR RECORD DOES NOT EXIST
*
         TM    IOBCSW+3,X'Ø2'            * UNIT CHECK ?
         BNO   TRYUEXC                   * NO, SO TRY UNIT EXCEPTION
*
         TM    IOBSENS1,X'Ø8'            * 'NO RECORD FOUND' ?
         BO    WSE                       * YES - WRITE SEQUENCE ERROR
         B     ERROR2Ø                   * ANY OTHER CASE IS FATAL ERROR
*
* TRACK FULL IS AN 'ACCEPTABLE' CONDITION, INDICATED BY UNIT EXCEPTION
*
```

```
TRYUEXC  EQU   *
         TM    IOBCSW+3,X'Ø1'            * UNIT EXCEPTION
         BO    EOT                       * YES - MEANS END OF TRACK
*
* UNIT CHECK PROBABLY MEANS HARD ERROR
*
         SR    R2,R2                     * R2 = ...
         ICM   R2,B'ØØ11',54(R9)         * ... CCW COUNT
         SR    R1,R1                     * R1 = ...
         ICM   R1,B'ØØ11',IOBCSW+5       * ... RESIDUAL COUNT
*
         SR    R2,R1                     * SAVE # BYTES WRITTEN ...
         ST    R2,4(R5)                  * ... FOR CALLER
         B     ERROR2Ø                   *     ... AND SIGNAL FATAL
ERROR
*
*-----------------------------------
* RECORD WRITTEN OK.  UPDATE LAST RECORD ID IN DCB.
*-----------------------------------
*
WRITEOK  EQU   *
         L     R11,ADCB                  * DCB ADDRESS ...
         USING IHADCB,R11                * ... AND ADDRESSABILITY
*
         MVC   DCBFDAD,IOBSEEK           * UPDATE DCBFDAD ...
         L     R1,IREC                   * ... TO POINT TO ...
         STC   R1,DCBFDAD+7              *     ... LAST RECORD WRITTEN
*
         SR    R15,R15                   * SET RETURN CODE
         B     RETURN                    * AND RETURN
*
         DROP  R1Ø,R11                   * FINISHED WITH IOB, DCB
         EJECT
*
*********************************************************************
* CLOSE THE DATASET
*********************************************************************
*
CLOSE    EQU   *
         L     R11,ADCB                  * R11 = DCB ADDRESS
*
         CLOSE ((R11)),MODE=31           * CLOSE DCB
*
         SR    R15,R15                   * CLEAR RETURN CODE
         EJECT
*
*********************************************************************
* RETURN TO CALLER
*********************************************************************
*
RETURN   EQU   *
         ST    R15,Ø(R7)                 * STORE IOSTAT FOR CALLER
         L     R13,4(R13)                * RESTORE ADDRESS OF HSA
```

```
         L     R14,12(R13)               * RESTORE R14
         LM    RØ,R12,2Ø(R13)            * RESTORE RØ-R12
         BR    R14                       * AND RETURN
         EJECT
*
*********************************************************************
* ERROR CONDITIONS
*********************************************************************
*
ERROR4   EQU   *
         LA    R15,4                     * OPEN FAILURE
         B     RETURN                    * RETURN
*
WSE      EQU   *
         LA    R15,8                     * WRITE SEQUENCE ERROR
         B     RETURN                    * RETURN
*
EOT      EQU   *
         LA    R15,12                    * TRACK OVERFLOW
         B     RETURN                    * RETURN
*
EOD      EQU   *
         LA    R15,16                    * EXTENT VIOLATION
         B     RETURN                    * RETURN
*
ERROR2Ø  EQU   *
         LA    R15,2Ø                    * I/O ERROR
         B     RETURN                    * RETURN
         EJECT
*
*********************************************************************
* CONSTANTS, VARIABLES AND DATA AREAS
*********************************************************************
*
         DS    ØD
         DC    CL8'SAVEAREA'
SAVEAREA DS    18F
SAVE913  DS    5F
*
AØSCR1   DS    A                         * @(SECTOR CONVERT ROUTINE)
APCNVT   DS    A                         * @(REAL BLOCK ADDRESS RTN)
TTRZ     DC    XL4'ØØØØØØØØ'             * TTRZ OF REQUESTED RECORD
*
ADCB     DC    A(DCB)
ADEB     DC    A(Ø)
AIOB     DC    A(IOBAREA)
ACHPROG  DC    A(CHPROG)
*
IREC     DC    F'Ø'
*
         DS    ØF
DCB      DCB   DDNAME=DUMMY,DSORG=PS,MACRF=E
```

```
LDCB     EQU   *-DCB
*
         DS    ØF
IOBAREA  DC    (LIOB)X'ØØ'
*
         DS    ØD
CHPROG   EQU   *
SETSECT  CCW   X'23',SECNUM,X'4Ø',1      * SET SECTOR
SEARCHID CCW   X'31',Ø,X'4Ø',5           * SEARCH ID EQUAL
TIC      CCW   X'Ø8',SEARCHID,Ø,Ø        * RETRY UNTIL SUCCESSFULL
WCKD1    CCW   X'1D',COUNT,X'8Ø',8       * WRITE COUNT, KEY, DATA
WCKD2    CCW   X'1D',IDAW,X'44',Ø        * WRITE COUNT, KEY, DATA
READSECT CCW   X'22',SECNUM,X'ØØ',1      * READ SECTOR
*
COUNT    DS    D                         * INDIRECT DATA ADDRESS WORD
IDAW     DS    F                         * INDIRECT DATA ADDRESS WORD
SECNUM   DC    XL1'ØØ'                   * SECTOR NUMBER
LCHPROG  EQU   *-CHPROG
*
IOB      DSECT
IOBFLAG1 DS    XL1                       * CHAINING/UNRELATED BITS
IOBFLAG2 DS    XL1                       * NOT USED HERE
IOBSENSE DS    ØXL2                      * SENSE BYTES
IOBSENSØ DS    XL1                       *   SENSE BYTE 1
IOBSENS1 DS    XL1                       *   SENSE BYTE 2
IOBECBCC DS    XL1                       * FIRST BYTE OF COMP. CODE
IOBECBPT DS    AL3                       * ECB ADDRESS
IOBFLAG3 DS    XL1                       * SYSTEM USE ONLY
IOBCSW   DS    XL7                       * CHANNEL STATUS WORD
IOBSIOCC DS    XL1                       * START SUBCHANNEL DATA
IOBSTART DS    AL3                       * CHANNEL PROGRAM ADDRESS
         DS    XL1                       * RESERVED
IOBDCBPT DS    AL3                       * DCB ADDRESS
IOBRESTR DS    XL1                       * USED FOR ERROR RECOVERY
         DS    XL3                       * USED FOR ERROR RECOVERY
IOBINCAM DS    XL2                       * USED FOR MAG TAPE ONLY
         DS    XL2                       * RESERVED
IOBSEEK  DS    XL8                       * SEEK ADDRESS (MBBCCHHR)
*
IOBECB   DS    F                         * ECB
LIOB     EQU   *-IOB
*
         PRINT NOGEN
         CVT   DSECT=YES                 * CVT MAPPING MACRO
         PRINT NOGEN
         DCBD  DSORG=PS,DEVD=DA          * DCB MAPPING MACRO
*
         END
```

P R S Wright
Associate Consultant
Tessella Support Services plc (UK)

Timed job submission

THE PROBLEM

These days, most batch systems tend to be controlled through some form of proprietary job scheduling system. Such facilities are the ideal way to control when jobs are run. However, they are often not the most practical way for users to ensure that *ad hoc* jobs are run at particular points in the nightly runs – for example, running some form of special one-off DB2 housekeeping job for the DBAs when there is less activity on the system, etc. Doing such a job through a controlled schedule can involve lengthy change-control mechanisms.

THE SOLUTION

As a result, the following system was put together. It is an edit macro that should be used instead of SUBMIT. It works in the following way:

- The user ensures that the job to be submitted is set up with a jobcard that has TYPRUN=HOLD on it.
- The edit macro is then issued with a time in the form HH.MM, where the hours and minutes equate to the number of hours and minutes after midnight on the day on which the job is submitted that it is required to run. For example to run a job at 1am in the morning, you would issue the command SUBAT 25.00 (where SUBAT is the name given to the edit macro). This curious form of time is neccesary because SUBAT uses JES2 timed commands to achieve its objective, and this is the way that these commands know time.
- The edit macro works out the special JES2 £TA command to be issued to release the job at the specified time, and then calls a home grown REXX function to issue the command. Note that this routine will not contravene any OPERCMDS settings in RACF and, if you want to use this command, it is necessary to ensure that users are allowed to issue the appropriate commands. Please note also that it is necessary to have a means of dynamically switching on APF through an SVC, though if this is lacking, I have provided

some example code (see the end of this article). And that's all there is to it.

At the moment the supplied command is set up to allow a maximum time of 6am the following morning, but because £TA can cover up to 99 hours, this can easily be changed. Note that the timed commands are lost around a re-start of JES2.

SUBAT

```
/* REXX */
/* */
/* Submit a job and release it at a specific time */
/* */
'ISREDIT MACRO (when)'
'ISREDIT F TYPRUN=HOLD ALL'
/* */
/* ensure there is a hold statement in the job */
/* */
IF RC¬=Ø THEN DO
   ZEDSMSG='TYPRUN=HOLD not found'
   ZEDLMSG='Job will not wait for command'
   'ISPEXEC SETMSG MSG(ISRZØØ1)'
   EXIT
   END
/* */
/* now validate the supplied time */
/* furthermore it must be later   */
/* than now */
/* */
PARSE VAR when hour '.' minute
IF hour> 3Ø | minute>6Ø | hour<Ø | minute<Ø THEN DO
   ZEDSMSG='Time should be HH.MM'
   ZEDLMSG='enter a time in format HH.MM'
   'ISPEXEC SETMSG MSG(ISRZØØ1)'
   EXIT
   END
check_hour=LEFT(TIME(),2)
check_MIN=SUBSTR(TIME(),4,2)
IF check_hour > hour | (check_hour = hour & check_MIN>=minute) THEN DO
   ZEDSMSG='Time too early'
   ZEDLMSG='must enter a later time'
   'ISPEXEC SETMSG MSG(ISRZØØ1)'
   EXIT
   END
/* */
/* to avoid having to save the dataset for a submit */
/* allocate a temporary file and copy this member   */
/* across. then submit that for execution           */
/* */
```

```
'ISREDIT (dsname) = DATASET'
CALL MSG(OFF)
'FREE FI(SPONGE)'
CALL MSG(ON)
user=SYSVAR(SYSUID)
“ALLOC FI(SPONGE) CATALOG DELETE DA(“user”.your.dsname)”,
   “DSORG(PS) REC(F B) LR(8Ø) BLK(2792Ø)”,
   “SPACE(1,2) TRACKS”
'ISREDIT (start) = LINENUM .ZF'
'ISREDIT (endit) = LINENUM .ZL'


DO cnt=start TO endit
'ISREDIT (line) = LINE' cnt
 QUEUE line
 END
QUEUE ''
'EXECIO * DISKW SPONGE (FINIS'
/* */
/* now submit the actual job and trap the job number for the */
/* timed command.                                            */
/* */
CALL OUTTRAP('line.')
'SUBMIT' user'.your.dsname'
do linecnt=1 to line.Ø
   jobloc=WORDPOS('JOB',line.linecnt)
   subloc=WORDPOS('SUBMITTED',line.linecnt)
   SAY line.linecnt
   IF jobloc¬=Ø & subloc¬=Ø THEN LEAVE
END
IF jobloc=Ø THEN DO
   ZEDSMSG='Job submission failure'
   ZEDLMSG='Check source for possible missing job card'
   'ISPEXEC SETMSG MSG(ISRZØØ1)'
   EXIT
   END
check_line=SUBWORD(line.linecnt,jobloc,2)
PARSE VAR check_line  . . '(' number ')' .
literal=”T A,T=”when”,'A” number”'”
/* */
/* now issue the command via REXWTO */
/* */
CALL REXWTO literal
exit
```

REXWTO

```
************************************************************
* REXWTO: A REXX DESIGNED TO ISSUE A COMMAND INTO THE SYSTEM
*
```

```
* TO OPERATE
* CALL REXWTO COMMAND
* WHERE COMMAND IS THE MVS COMMAND (UP TO 8Ø CHARACTERS LONG).
*
* IF THE COMMAND IS ISSUED, THE RC=Ø WILL BE SET, ELSE RC=99.
*
* NOTE THAT NO ACCOUNT IS TAKEN OF OPERCMDS IN RACF SUBSEQUENTLY
* DENYING THE COMMANDS OPERATION.
************************************************************
         MACRO
         REXREGS
         LCLA &CNT
&CNT     SETA Ø
.LOOP    ANOP
R&CNT    EQU &CNT
&CNT     SETA &CNT+1
         AIF (&CNT LT 16).LOOP
         MEND
REXWTO   TITLE 'REXX FUNCTION TO ISSUE MVS COMMANDS'
REXWTO   AMODE 31
REXWTO   RMODE ANY
REXWTO   CSECT
SVCAUTH  EQU 235    * <=== set to your apf on svc number
SVCDAUTH EQU 236    * <=== set to your apf off svc number
         REXREGS
         BAKR  14,Ø
         LR    12,15
         USING REXWTO,12
*
         LR    R1Ø,RØ                  * R1Ø -> A(ENVIRONMENT BLOCK)
         USING ENVBLOCK,R1Ø
*
         LR    R11,R1                  * R11 -> A(PARAM LIST (EFPL))
         USING EFPL,R11
*
         L     R9,ENVBLOCK_IRXEXTE     * R9 -> A(EXTERNAL EP TABLE)
         USING IRXEXTE,R9
*
         L     R6,EFPLARG              * R6 -> A(ARGUMENT TABLE)
         USING ARGTABLE_ENTRY,R6
         L     R7,EFPLEVAL
         L     R7,Ø(R7)                *R7 -> A(EVALUATION BLOCK
         USING EVALBLOCK,R7
*
         STORAGE OBTAIN,LENGTH=COMSLEN,ADDR=(8)
         USING COMSDS,R8


*
* PREPARE THE REXX AREA FOR USE
*
         XC    COMS(COMSLEN),COMS      * SET TO LOW VALUES
```

```
         LA    15,COMID
         ST    15,COMS
         LA    15,COMDUMMY
         ST    15,COMS+4
         ST    15,COMS+8
         LA    15,COMSHVB
         ST    15,COMS+12
         LA    15,COMRET
         ST    15,COMS+16
         OI    COMS+16,X'8Ø'           * INDICATE END OF PARMS
         MVC   COMID,=C'IRXEXCOM'
         MVC   RCØ,=C'ØØ'
*
         CLC   ARGTABLE_ARGSTRING_PTR(8),=2F'-1' *END OF ARGS?
         BE    RCNOK              * YES SO STRAIGHT OUT
*
         L     R2,ARGTABLE_ARGSTRING_PTR    * R2 -> A(ARGUMENT)
         L     R1,ARGTABLE_ARGSTRING_LENGTH *R1 -> L(ARGUMENT)
*
* THE SUPPLIED VARIABLE CANNOT BE MORE THAN 8Ø CHARACTERS IN LENGTH.
* NOR CAN IT BE ZERO.
*
         C     R1,=F'8Ø'                    *IS THE LENGTH GT 8Ø
         BH    RCNOK                        * YES, SO IGNORE
         C     R1,=F'Ø'                     *IS THE LENGTH Ø
         BNH   RCNOK                        * YES, SO IGNORE
         BCTR  R1,Ø
         EX    R1,WTOMOVE
EXLOCT   LOCTR
WTOMOVE  MVC   WTO(Ø),Ø(R2)                 * YES, STORE COMMAND
REXWTO   LOCTR
         SVC SVCAUTH
         MODESET MODE=SUP,KEY=ZERO
         MVC   COMMAND(4),=X'ØØ54ØØØØ'
         OC    WTO,BLANKS
         XR    RØ,RØ                CLEAR RØ FOR SVC 34
         LA    R1,COMMAND           SET R1 TO COMAND LOCATION.
         SVC   34
         MODESET MODE=PROB
         SVC  SVCDAUTH
         B     SETRCØ
RCNOK    DS ØH
         MVC RCØ,=C'99'
SETRCØ   DS    ØH        * NOW RETURN THE INFO.
*
         LA    R6,COMSHVB
         USING SHVBLOCK,R6
         ST    R1Ø,COMRET


         XC    COMSHVB(SHVBLEN),COMSHVB
         MVI   SHVCODE,C'S'
         XC    SHVNEXT,SHVNEXT
```

```
RCOK     DS ØH
         MVC   EVALBLOCK_EVLEN,=F'2'
         MVC   EVALBLOCK_EVDATA(2),RCØ
         LA    R5,RC
         ST    R5,SHVNAMA
         LA    R5,2
         ST    R5,SHVNAML
         LA    R5,RCØ            * ADDRESS WTO PROFILE
         ST    R5,SHVVALA
         LA    R5,2              * GET PROFILE LENGTH
         ST    R5,SHVVALL
         LR    RØ,R1Ø
         LA    R1,COMS
         L     R15,IRXEXCOM
         BALR  R14,R15
         BZ    ENDREXX
         ABEND 1
ENDREXX  DS ØH
         STORAGE RELEASE,LENGTH=COMSLEN,ADDR=(8)
         PR
         LTORG
BLANKS   DC    8ØC' '
RC       DC    C'RC'
*********************************************************************
***      IRXEXCOM PARAMETER AREA                                  ***
*********************************************************************
*
COMSDS   DSECT
COMS     DS    5AL4
COMID    DS    CL8              * IRXEXCOM ID - C'IRXEXCOM'
COMDUMMY DS    AL4              * NOT USED
COMSHVB  DS    (SHVBLEN)X       * IRXEXCOM SHVBLOCK (LENGTH FROM DSECT)
COMRET   DS    AL4              * IRXECOM RC
RCØ      DS    CL2
COMMAND  DS    F
WTO      DS    CL8Ø
COMSLEN  EQU *-COMS
         DS ØD
         IRXEFPL
         IRXARGTB
         IRXEVALB
         IRXENVB
         IRXEXTE
         IRXSHVB
         END
```

AUTOMATIC SVC LOADER WITH ASSOCIATED ON/OFF APF SVCS

Remember to link this loader with AC(1) and place in an APF authorized library.

```
         MACRO
         REXREGS
         LCLA &CNT
&CNT     SETA Ø
.LOOP    ANOP
R&CNT    EQU &CNT
&CNT     SETA &CNT+1
         AIF (&CNT LT 16).LOOP
         MEND
         TITLE 'AUTOAUTH - INITIALIZATION'
         PRINT NOGEN
AUTOAUTH AMODE 31
AUTOAUTH CSECT
         REXREGS
*
         BAKR R14,RØ
*
NEWSVC1  EQU 235     < === set to the apf on svc
NEWSVC2  EQU 236     < === set to the apf off svc
*
         LR R12,R15
         USING AUTOAUTH,R12
*
*** THIS MODULE WILL DYNAMICALLY ADD AN AUTHORIZE AND A DE-AUTHORIZE
*** SVC TO THE SVCTABLE AS DEFINED BY NEWSVC1 AND NEWSVC2
*
         TITLE 'LOAD CSA CODE'
         MODESET MODE=SUP,KEY=ZERO   GET INTO SUPERVISOR STATE
         LOAD EP=AUTOSVC,LOADPT=LOADPT
         LR  R5,RØ       * GET THE ENTRY ADDRESS
*                         * THIS WILL ALLOW THE ADDRESSING MODE
*                         * TO BE DETERMINED
         SRL R5,31       * GET RID OF ALL THE EXCESS 'BITS'
         SLL R5,31       * AND PUT IT BACK WHERE IT WAS FOR AN OR.
*
*** NOW LETS GET THE MODULE SIZE FOR A SYSTEM BASED GETMAIN.
*
         SLL R1,8        * DROP AUTH INFO
         SRL R1,5        * GET ACTUAL SIZE
         LR  R3,R1       * AND PLACE IN A USABLE REGISTER.
         STORAGE OBTAIN,LENGTH=(3),OWNER=SYSTEM,ADDR=(2),SP=241
         LR  R4,R2       * KEEP THE ADDRESS OF THE STORAGE
         OR  R4,R5       * SET THE ADDRESSING MODE.
         LR  R11,R3      * SET UP LENGTHS FOR MVCL
         L   R1Ø,LOADPT  * RELOAD THE LOAD POINT
         MVCL R2,R1Ø
         LA    R8,MAJENQ            *


         LA    R9,MINENQ            * FORCE SERIALIZATION
         ENQ   ((8),(9),E,8,SYSTEM)  *
         SVCUPDTE NEWSVC1,REPLACE,TYPE=4,EP=(4)
*
```

```
         LA    R8,MAJENQ
         LA    R9,MINENQ             * FORCE SERIALIZATION
         DEQ   ((8),(9),8,SYSTEM)    * DEQUEUE ROUTINE
*
*** NOW REPEAT FOR THE DE-AUTHORISE SVC
*
         LOAD EP=AUTOSVC1,LOADPT=LOADPT
         LR  R5,RØ       * GET THE ENTRY ADDRESS
*                        * THIS WILL ALLOW THE ADDRESSING MODE
*                        * TO BE DETERMINED
         SRL R5,31       * GET RID OF ALL THE EXCESS 'BITS'
         SLL R5,31       * AND PUT IT BACK WHERE IT WAS FOR AN OR.
*
*** NOW LETS GET THE MODULE SIZE FOR A SYSTEM BASED GETMAIN.
*
         SLL R1,8        * DROP AUTH INFO
         SRL R1,5        * GET ACTUAL SIZE
         LR  R3,R1       * AND PLACE IN A USABLE REGISTER.
         STORAGE OBTAIN,LENGTH=(3),OWNER=SYSTEM,ADDR=(2),SP=241
         LR  R4,R2       * KEEP THE ADDRESS OF THE STORAGE
         OR  R4,R5       * SET THE ADDRESSING MODE.
         LR  R11,R3      * SET UP LENGTHS FOR MVCL
         L   R1Ø,LOADPT  * RELOAD THE LOAD POINT
         MVCL R2,R1Ø
         LA    R8,MAJENQ             *
         LA    R9,MINENQ             * FORCE SERIALIZATION
         ENQ   ((8),(9),E,8,SYSTEM) *
         SVCUPDTE NEWSVC2,REPLACE,TYPE=4,EP=(4)
*
         LA    R8,MAJENQ
         LA    R9,MINENQ              FORCE SERIALIzATION
         DEQ   ((8),(9),8,SYSTEM)    DEQUEUE ROUTINE
         LA    R15,Ø
         PR
MAJENQ   DC    CL8'SYSZSVC'
MINENQ   DC    CL8'TABLE'
LOADPT   DS    F
         LTORG
         PRINT NOGEN
         IHAPSA
         CVT    DSECT=YES
         IHASCVT
*
         END
```

APF ON SVC

```
AUTOSVC  CSECT                   ADDR
AUTOSVC  AMODE 31
         USING *,6
         USING TCB,4
         L    3,TCBJSCB     * ADDRESS THE JSCB
```

```
         USING IEZJSCB,3
         OI JSCBOPTS,JSCBAUTH
         BR 14
         LTORG
         PRINT NOGEN
         IKJTCB
         IEZJSCB
         END
```

APF OFF SVC

```
AUTOSVC1 CSECT
AUTOSVC1 AMODE 31
         USING *,6
         USING TCB,4
         L    3,TCBJSCB     * ADDRESS THE JSCB
         USING IEZJSCB,3
         NI JSCBOPTS,X'FF'-JSCBAUTH
         BR 14
         LTORG
         PRINT NOGEN
         IKJTCB
         IEZJSCB
         END
```

C A Jacques
Systems Programmer (UK)

A REXX utility to delete PDS members

THE PROBLEM

On a number of occasions, I have had to delete a series of PDS members – sometimes based on dates, sometimes based on user-ids and sometimes by member names. You can imagine the effort required to delete each member one by one by going through option 3.1!

THE SOLUTION

The following utility deletes PDS members with a simple command and without going through option ISPF option 3.1. With this utility,

you can delete members by:

- Name (one member only). Examples of names are (UPDATE) and (VMUPDATE).
- Range of names through a pattern by using a wild character (only one wild character is allowed). Examples of pattern are (MVS*), (M*E), (*UPDATE), and (*).
- User-id (both equal to and not equal to are supported). Examples of userids are, ID=DORMANT ID<>ISNOMORE ID=" ID<>".
- Creation date (less than, greater than or equal to are supported). Example of this usage CRE<89/02/18, CRE>95/05/31, CRE=97/01/01.
- Changed date (less than, greater than or equal to are supported). Example of this usage CHA<97/01/01, CHA>95/05/31, and CHA=97/01/01.

The full format to call this utility is:

```
{TSO} XPRSDEL Pdsname{(MemPattern)} {Criteria}
```

where Pdsname is the input PDS name. Follow TSO rules (ie if the name doesn't start with a quote, your user-id will be suffixed to the dataset name).

MemPattern is either a single name of the PDS member or a pattern. Use a single '*' as a wild character. It can be prefixed, suffixed or in the middle. To select the whole PDS, either use * or just drop the entire MemPattern. There is a subtle difference between the two which is explained later.

```
Criteria is {ID=xxxxxxxx ] ID<>xxxxxxx} {&}{]}
            {CRE<YY/MM/DD ] CRE>YY/MM/DD ] CRE=YY/MM/DD} {&}{]}
            {CHA<YY/MM/DD ] CHA>YY/MM/DD ] CHA=YY/MM/DD}
          Or
            {ID=' '}
```

and the simplest format is:

```
{TSO} XPRSDEL
```

You can call this utility either from ISPF option 6 or by appending TSO to this command from anywhere in ISPF (except in SDSF).

Also criteria and PDS name can be in any order.

If you call this utility in its simple format, the program will prompt you for the PDS name.

If the input is not a PDS, the program will display a message and abort.

If you entered just the PDS name or pass the PDS name as a parameter (ie without any member name or pattern) the program will delete all members created by you (before the start of the deletion process, it will prompt you for confirmation). If you use a pattern – including (*) with a PDS name, the program will ask you whether you want to delete members created or last modified by others too (ie it will delete all members created by anyone that meets the selection criteria).

You can limit your selection by passing the criteria as a parameter.

Criteria can be used only if the PDS has ISPF statistics stored and can only be passed as a parameter, ie the program will not prompt you for criteria! Criteria can be a combination of any of the following three parameters:

- ID is the user-id who created/last modified the member.
- CRE – the creation date in YY/MM/DD format.
- CHA – the date of last modification in YY/MM/DD format.

For ID parameter, you can use the following comparators:

- = Equal To
- <> Not Equal To

For example: ID='NOMORE' will select all the members created/ modified by NOMORE; ID<>'HASLEFT' will select all the members that were not created/modified by HASLEFT.

ID='' will select all the members that have no user-id stored in the ISPF PDS statistics. Note:

- If you use ID='' or ID<>'', then do not use CRE and CHA parameters along with it.

- Do not use quotes in criteria except for ID=" and ID<>"
- You can use \= instead of <>.

For CRE & CHA parameters, the following comparators are used:

- < Less than
- > Greater than
- = Equal to

For example: CRE<87/01/30 will select all the members created before 87/01/30; CHA=96/05/02 will select all the members modified on 96/05/02; and CHA>97/03/04 will select all the members modified after 97/03/04.

You can have a combination of ID, CRE, and CHA by using & (AND) or | (OR).

For example: (ID=HASLEFT)&(CRE<93/03/20) will delete those from the selection that were created/modified by HASLEFT before a creation date of 93/03/20.

Note:

- Only one ID, CRE, and CHA parameter is allowed in a single call.
- No validation is done against dates!
- No validation is done against parameters, ie If you type IDS instead of ID, no validation is performed. However, because of this error, no members will pass the criteria.

The program creates a dataset that has a list of members deleted from the PDS for audit purposes and displays the name of this dataset at the end of the program.

There are a number of defaults you can modify:

- You can modify the name of the audit dataset. Currently it is set to DELETED.MEMBERS.LIST.
- If you do not want to have the above audited dataset and would

like to see the list of members that are being deleted and those members that are not passing the criteria, if criteria have been passed, on the terminal set Watch to 'Y'. Currently it has been set to 'N'.

- By default, the program will not delete members from the PDS created by others if no criteria have been passed. If you want this program to delete all members that satisfy the member name pattern, set Others to 'Y'. It is set to 'N' in the program.

All the defaults are in the BuildDflts subroutine. Please modify it to suit your installation standards.

One final note: if the ISPF statistics do not store the CREation date and CHAnged date in YY/MM/DD format, modify the compare logic present in CheckCriteria subroutine suitably. I have inserted comments there for your easy reference/modifications. You can pass? or HELP as a parameter to display help on how to use this REXX program. When you start this program, it displays the DATAID allocated by the ISPF PDF services. In case you have to cancel the program while it is running or the program aborts (if the job ends normally, a message is displayed), you have to close/free the PDS. If you don't close/free the PDS in such instances, the next time you access the PDS, you will get a return code of 8. To remedy such situations either log-off from ISPF and log-on or call 'CLOSEPDS' EXEC to close the PDS. You can pass the Dataid as a parameter or let the REXX program prompt for it. The code for CLOSEPDS is shown below.

CLOSEPDS

```
/* ———————— REXX ———————————— */
/* REXX EXEC To Close a dataset with DataId                */
/* ———————— REXX ———————————— */
Arg xDataId .
If xDataId='' Then Do
  Say 'Type The Dataid and Press Enter'
  Pull xDataId
  End
'ISPEXEC LMCLOSE DATAID('xDataId')'
'ISPEXEC LMFREE  DATAID('xDataId')'
Exit
```

XPRSDEL

```
/* ———————— REXX————————————— */
/* REXX EXEC To Delete Members From a PDS                    */
/* ———————————— REXX ————————————— */
/* Format is                                                 */
/*   <TSO> XPRSDEL Pdsname(pattern) <Criteria>               */
/* Pattern *SAM MAK* MOY*FIR                                 */
/* Criteria Can Be A Combination Of                          */
/*  ID CRE & CHA                                             */
/* Ex ID=MOYEENKH&CHA<97/Ø2/Ø1                               */
/* ——————————————————————————— */
/* Naming Conventions Used:-                                 */
/* REXX Commands & Functions start with a Capital letter     */
/* TSO  Commands are in Upper Case                           */
/* User data (Variables/Constants/Labels) is a combination */
/* Of Upper & Lower case and                                 */
/*  Alphanumeric user data starts with x  (lower x)          */
/*  Numeric      user data starts with n  (lower n)          */
/*  Switches / Flags       start  with s  (lower s)          */
/*  Condition (True/False) start  with c  (lower c)          */
/*  REXX Labels Start with a Capital Letter                  */
/* ——————————————————————————— */

Trace O
Arg xParams
Select
 When xParams='' Then Do
  xFromPds=''
  xCriteria=''
  End
 When xParams='?' | xFromPds='HELP' Then Do
  Call How2Use
  Exit
  End
 Otherwise Call SplitParams
End
Call BuildDflts                          /* Defaults are Here !!!    */
Call InitVarbls
If xFromPds='' Then Call GetInpPds
Call CheckInpPds
If xCriteria='' Then Call GetSomeDflts
Call Start2Del
Call EojInfo
Exit

/*———————  Start Of Subroutines ——————————*/

GetInpPds:
'CLRSCRN'
Say Center('Type The Input File Name - TSO Rules Apply',72)
```

```
Say Center('ie If The Name Is Not In Quotes, 'Userid() 'Will Be
Suffixed',72)
Say Center('————————————————————————————————',72)
Say Center('Input Can Only Be A PDS',72)
Say Center('Type Just The PDS Name To Delete All Members Created/
Modified By' Userid(),72)
Say Center('Choose A Pattern For Partial Selection - Use * As A Wild
Character',72)
Say Center('Note:- Only One Wild Character Allowed For Partial
Selection',72)
Say Center('Example: Your Selection Can Be  SA*ENA OR NB* OR *ASK Or
*',72)
Pull xFromPds .
xFromPds=Strip(Translate(xFromPds,"'",'"'))
Select
 When xFromPds='' Then Do
  Say '*Error* You Have To Type A PDS Name '
  Say '        Program Aborted'
  Exit
  End
 When Left(xFromPds,1)<>"'" Then Do
  xFromPds="'"Userid()'.'xFromPds"'"
  End
 When Left(xFromPds,1)="'" & Right(xFromPds,1)<>"'" Then Do
  xFromPds=xFromPds"'"
  End
 Otherwise Nop
 End
Return

CheckInpPds:
xMemPat=''
If Left(xFromPds,1)<>"'" Then xFromPds="'"Userid()'.'xFromPds"'"
If Pos('(',xFromPds)>Ø Then Do
  Parse Var xFromPds  xFromPds '(' xMemPat ')' .
  If Verify(xMemPat,'*')=Ø Then xMemPat=''
  xFromPds=xFromPds"'"
  sWholePds='N'
  End
 Else sWholePds='Y'
xAvail=SYSDSN(xFromPds)
If xAvail<>'OK' Then Do
  Say '*Error*' xFromPds 'Does Not Exist'
  Say '        Program Aborted'
  Exit
  End
xAvailRc=LISTDSI(xFromPds NORECALL)
If Rc<4 Then Do
  If Left(SYSDSORG,2)='PO' Then Nop
    Else Do
     Say '*Error* The Data Set Organization Of' xFromPds
     Say '        Is Not Supported By This REXX'
```

```
      Say '        Program Aborted'
      Exit
      End
  End
Return

GetSomeDflts:
If sWholePds='Y' Then Do
  Say '*Note* You Have Selected To Delete All Members From' xFromPds'.'
  Say '       ALL Members Created/Modified By You Will Be Deleted.'
  Say '       Please Confirm By Typing Y'
  Pull xAnswer .
  If xAnswer<>'Y' Then Exit
  End
 Else Do
  xAns.Y='Yes'
  xAns.N='No'
  Say '       Do You Want To Delete Members Created/Modified By Others?'
  Say '       Type YES or NO   OR'
  Say '       Press Enter To Accept The Default' xAns.sOthers
  Pull sOthers .
  If Abbrev('YES',sOthers,1) Then sOthers='Y'
   Else sOthers='N'
  End
Return

Start2Del:
If Pos('*',xMemPat)>Ø Then Do
  Do nI=1 By 1 Until nI=Length(xMemPat)
   If Substr(xMemPat,nI,1)='*' Then nStars=nStars+1
   End
  nLimit=nStars>1
  End
Select
 When sWholePds='Y' Then Nop
 When xMemPat='' Then Nop
 When Pos('*',xMemPat)=Ø Then Do
  sJust1Mem='Y'
  xMemName=xMemPat
  End
 When nLimit>Ø  Then Nop
 When Left(xMemPat,1)='*' Then Do
  xSufx=Strip(Translate(xMemPat,'','*'))
  nLimit=Words(xPrfx)>1
  End
 When Right(xMemPat,1)='*' Then Do
  xPrfx=Strip(Translate(xMemPat,'','*'))
  xMemName=xPrfx
  nLimit=Words(xSufx)>1
  End
 Otherwise Do
  xTemp=Strip(Translate(xMemPat,'','*'))
```

```
  nLimit=Words(xTemp)>2
  Parse Var xTemp xPrfx xSufx .
  xMemName=xPrfx
  End
 End
If nLimit>Ø Then Do
  Say '*Error* Invalid Selection Pattern Found - Pattern = ' xMemPat
  Say 'Program Aborted'
  Exit
  End
'ISPEXEC LMINIT DATAID(xDataId) DATASET('xFromPds') ENQ(SHRW)'
Say
Say 'Please Note the Dataid for the PDS ' xDataId
Say
If Rc<>Ø Then Do
  Say 'Unable to Access' xFromPds 'RC='Rc
  Exit
  End
'ISPEXEC LMOPEN DATAID('xDataId') OPTION(OUTPUT)'
If Rc<>Ø Then Do
  Say 'Unable to Open' xFromPds 'RC='Rc
  Exit
  End
Do nI=1
 'ISPEXEC LMMLIST DATAID('xDataId') OPTION(LIST) MEMBER(xMemName)
STATS(YES)'
 Select
  When Rc<>Ø Then Do
   If nI=1 Then Do
     Say '*Error*' xFromPds 'Is an Empty Dataset!'
     Say '        Program aborted'
     sFatalErr='Y'
     End
   Leave nI
   End
  When xPrfx<>'' & Left(xMemName,Length(xPrfx))<>xPrfx Then Leave nI
  Otherwise Nop
  End
 xMemName=Strip(xMemName)
 sThisMem='N'
 xSaveCrt=''
 If xCriteria<>'' Then Call CheckCriteria
 Select
       /* Failed Criteria - Don't bother further               */
  When xCriteria<>'' & sPassCriteria='N' Then Nop

       /* Passed Criteria, - Just 1 Member                     */
       /* Do whatsoever and prepare to leave!                  */
  When sJust1Mem='Y' Then Call OneMemOnly

       /* Full Pds and Passed criteria  - Member Passed        */
  When xMemPat='' & xCriteria<>'' Then sThisMem='Y'
```

```
       /* Full Pds But No Criteria - If myself, Member Passed  */
  When xMemPat='' & zluser=Userid() Then sThisMem='Y'

       /* Full Pds But No Criteria - Take all Members          */
  When xMemPat='' & sOthers='Y'     Then sThisMem='Y'

       /* Pattern Present - No Criteria - Not myself -Ignore   */
  When sOthers='N' & xCriteria=''  & zluser<>Userid() Then Do
   sThisMem='N'
   nOthrMems=nOthrMems+1
   End

       /* Both Suffix & Prefix Given                          */
  When xSufx<>'' & xPrfx<>'' Then Do
   If Right(xMemName,Length(xSufx))=xSufx &,
      Left(xMemName,Length(xPrfx))=xPrfx Then sThisMem='Y'
   End

       /* Only Suffix Given                                   */
  When xSufx <>'' & Right(xMemName,Length(xSufx))=xSufx Then
sThisMem='Y'

       /* Only Prefix Given                                   */
  When xPrfx <>'' & Left(xMemName,Length(xPrfx))=xPrfx Then sThisMem='Y
'

  Otherwise Nop
  End
 Select
  When sThisMem='Y' Then Nop
  When sTime2Leave='Y' Then Leave nI
  Otherwise Iterate nI
  End
 'ISPEXEC LMMDEL DATAID('xDataId') MEMBER('xMemName')'
 If Rc=Ø Then nDelMems=nDelMems+1
   Else Do
    Say '*Error* Fatal Error While Deleting Member' xMemName 'RC='Rc
    Say '        Program Abandoned'
    Leave nI
    End
 sDeletes='Y'
 If xSaveCrt<>'' Then xSaveCrt='('xSaveCrt')'
 xAudit.1=Left(xMemName,8) 'Deleted From' xFromPds xSaveCrt
 If sWatch='Y' Then Say xAudit.1
   Else Call RiteAudit
 If sJust1Mem='Y' Then Leave
 End
'ISPEXEC LMCLOSE DATAID('xDataId')'
'ISPEXEC LMFREE  DATAID('xDataId')'
Return
```

```
CheckCriteria:
xSaveCrt=xCriteria
xMsg.=''
If Pos('ID',xCriteria)>Ø Then Do
  nPos=Pos('ID',xSaveCrt)+2
  xComprtr=Substr(xSaveCrt,nPos,1)
  If Pos(xComprtr,'\=<>^')=Ø Then sThisMem='I'
   Else Do
    If zluser='' Then zluser="'"Copies(' ',8)"'"
    Parse Var xSaveCrt xPart1 'ID' xPart2
    xSaveCrt=xPart1||zluser||xPart2
    /* ---------------------------------*/
    /* Let us impersonate those users whose id contains CRE or CHA  */
    /* ---------------------------------*/
    If Left(xPart2,1)='=' Then xPart2=Substr(xPart2,2)
      Else xPart2=Substr(xPart2,3)
    If Verify(xPart2,')&|','M')>Ø Then Do
       nPos=Verify(xPart2,')&|','M')
       Parse Var xPart2 1 xPart3 =(nPos) .
       If Pos('CRE',xPart3)>Ø Then Do
          Parse Var xSaveCrt xPart1 'CRE' xPart2
          xSaveCrt=xPart1||'\'||xPart2
          End
       If Pos('CHA',xPart3)>Ø Then Do
          Parse Var xSaveCrt xPart1 'CHA' xPart2
          xSaveCrt=xPart1||'~'||xPart2
          End
       End
    End
  End
If Pos('CRE',xSaveCrt)>Ø Then Do
  nPos=Pos('CRE',xSaveCrt)+3
  xComprtr=Substr(xSaveCrt,nPos,1)
  If zlcdate<>'' & Pos(xComprtr,'\=<>^')>Ø Then Do
```

Editor's note: this article will be continued in the next issue.

Moyeen Ahmed Khan
Systems Programmer (Canada)

```
    /*————————Note———————————*/
    /* If zlcdate is not in yy/mm/dd format        */
    /*    uncomment and modify lines suitably      */
    /*————————Note———————————*/
    /* Parse Var zlcdate mm '/' dd '/' yy          */
    /* zlcdate=yy'/'mm'/'dd                        */
    /*————————Note———————————*/
    Parse Var xSaveCrt xPart1 'CRE' xPart2
    xSaveCrt=xPart1||zlcdate||xPart2
    End
   Else sThisMem='I'
 End
If Pos('CHA',xSaveCrt)>0 Then Do
  nPos=Pos('CHA',xSaveCrt)+3
  xComprtr=Substr(xSaveCrt,nPos,1)
  If zlmdate<>'' & Pos(xComprtr,'\=<>^')>0 Then Do
    /*————————Note———————————*/
    /* If zlmdate is not in yy/mm/dd format        */
    /*    uncomment and modify lines suitably      */
    /*————————Note———————————*/
    /* Parse Var zlmdate mm '/' dd '/' yy          */
    /* zlmdate=yy'/'mm'/'dd                        */
    /*————————Note———————————*/
    Parse Var xSaveCrt xPart1 'CHA' xPart2
    xSaveCrt=xPart1||zlmdate||xPart2
    End
   Else sThisMem='I'
 End
If Pos('~',xSaveCrt)>0 Then Do
  Parse Var xSaveCrt xPart1 '~' xPart2
  xSaveCrt=xPart1||'CHA'||xPart2
  End
If Pos('\',xSaveCrt)>0 Then Do
  Parse Var xSaveCrt xPart1 '\' xPart2
  xSaveCrt=xPart1||'CRE'||xPart2
  End
/*If sThisMem='N' Then Do*/
/*  Interpret 'cTrue='Translate(xSaveCrt,'!','/')*/
/*  If cTrue Then sThisMem='Y'*/
/*  End*/
/* Else sThisMem='N'*/
Interpret 'cTrue='Translate(xSaveCrt,'!','/')
Select
 When \Datatype(cTrue,'W') Then Do
  Say '*Error* Invalid Parameter passed ' xParams
  Say '        Do not enclose Criteria under quotes'
  sPassCriteria='N'
  sTime2Leave='Y'
  sFatalErr='Y'
  End
```

```
 When cTrue Then sPassCriteria='Y'
 Otherwise sPassCriteria='N'
 End
Return

OneMemOnly:
sWrongUser='N'
Select
 When xMemPat<>xMemName Then Do
  sDeletes='N'
  sTime2Leave='Y'
  End
 When sOthers='Y' Then Do
  sThisMem='Y'
  End
 When sOthers='N' & zluser=Userid() Then Do
  sThisMem='Y'
  End
 Otherwise Do
  sWrongUser='Y'
  sDeletes='N'
  sTime2Leave='Y'
  End
 End
Return

RiteAudit:
If nOutRecs=Ø Then Call AllocOut
'EXECIO 1 DISKW xOutDsn  (STEM xAudit.'
If Rc<>Ø Then Do
  Say '*Error* Write Failure On ' xFileOut 'RC='rc
  Say '        Program Aborted'
  xSamFir=MSG('OFF')
  'FREE DD(xOutDsn)'
  xAskNb=MSG(xSamFir)
  Exit 16
  End
nOutRecs=nOutRecs+1
Return

AllocOut:
xOutName="'"xOutName"'"
xAvail=SYSDSN(xOutName)
If xAvail='OK' Then Do
  xSamFir=MSG('OFF')
  'DELETE' xOutName
  xAskNb=MSG(xSamFir)
  If Rc<>Ø Then Do
    Say '*Error*' xOutName 'Could Not Be Deleted - Return Code = ' Rc
    Say 'Program Aborted'
    Exit
    End
```

```
  End
nLrecl=132
xDfltDisp='NEW UNIT(SYSDA) LRECL('nLrecl')',
          'SPACE(2Ø) DSORG(PS) RECFM(F,B) TRACKS  RELEASE'
xSamFir=MSG('OFF')
'FREE DD(xOutDsn)'
xAskNb=MSG(xSamFir)
'ALLOCATE DSN('xOutName') DD(xOutDsn)' xDfltDisp
If Rc<>Ø Then Do
  Say '*Error* Unable To Alloc' xOutName
  Say 'Return Code Is ' Rc
  Exit 16
  End
Return

EojInfo:
Select
 When sFatalErr='Y' Then Nop
 When sDeletes='N' & sWrongUser='Y' Then Do
  Say '*Note*'xMemPat 'Was Created/Modified By' zluser
  Say '        Program Could Not Delete This Member'
  End
 When sDeletes='N' & nOthrMems>Ø Then Do
  If xMemPat='' Then xMemPat='*'
  Say '*Warning* No Members Found In Pds' xFromPds 'That Could Match'
xMemPat
  Say '          And Created/Modified By' Userid()
  Say '          (However There Were' nOthrMems 'Members Created By
Others'
  Say '          Program Could Not Delete Any Members'
  End
 When sDeletes='N' & xMemPat='' Then Do
  Say '*Warning* No Members Found In Pds' xFromPds
  Say '          That Could Match' xCriteria
  Say '          ( Has the criteria been coded correctly? )'
  Say '          Program Could Not Delete Any Members'
  End
 When sDeletes='N' Then Do
  Say '*Warning* No Members Found In PDS' xFromPds 'That Could Match'
xMemPat
  Say '          Program Could Not Delete Any Members'
  End
 When sWatch<>'Y' Then Do
  Say '*Note* List Of Members Deleted is in' xOutName
  Say '       (Total Members Deleted='nDelMems')'
  End
 Otherwise Nop
 End
Say '*Note* Job terminated normally'
Return
```

```
InitVarbls:
nDelMems=Ø
nLimit=Ø
nOthrMems=Ø
nOutRecs=Ø
nStars=Ø
sDeletes='N'
sFatalErr='N'
sJust1Mem='N'
sPassCriteria=''
sTime2Leave='N'
xAudit.Ø=1
xPrfx=''
xSufx=''
Return

/* ---Note--------Note----------Note---- */
/* The Defaults are defined here. Modify them to suit your          */
/* installations standards.                                         */
/* ---Note--------Note----------Note---- */

BuildDflts:
xOutName=Userid()'.DELETED.MEMBERS.LIST'     /* Output Dataset Name  */
sWatch='N'                             /* Y will display delete mems  */
sOthers='N'                            /* Y = select All Members      */
                                       /* N = select My Members ONLY  */
Return

SplitParams:
xCriteria=''
xFromPds=''
Select
 When Words(xParams)=1 Then Do
  If Verify(xParams,'=<>','M')>Ø Then xCriteria=xParams
   Else xFromPds=xParams
  End
 When Words(xParams)=2 Then Do
  If Verify(Word(xParams,1),'=<>','M')=Ø Then Parse Var xParams xFromPds
xCriteria
   Else Parse Var xParams xCriteria xFromPds
  End
 Otherwise Do
  Say '*Error*' xParams 'Contains Invalid Parameters'
  Say '       Program Aborted'
  Exit
  End
 End
Return

How2Use:
'CLRSCRN'
Parse Source . . xExecName .
```

```
Queue ‘Use ‘ xExecName ‘To Delete Members From a Partition Dataset
(Pds)’
Queue ‘————————————————————’
Queue ‘Format of the Command is (Assuming that the Exec is in one of
your’
Queue ‘                          SYSEXEC Libraries)’
Queue ‘<TSO>’ xExecName ‘PdsName<Pattern> <Criteria>’
Queue ‘If you are not in ISPF option 6, then suffix the command with
TSO’
Queue ‘For PDS name, follow TSO rules. Place the PDS name in quotes if’
Queue ‘it is not starting with your Userid.’
Queue ‘Type just the PDS name to delete members created/modified by’
Userid() ‘OR’
Queue ‘Choose a single member OR’
Queue ‘Choose a pattern for partial selection - Use * as a wild
character’
Queue ‘Note:- Only one wild character allowed for partial selection’
Queue ‘Example: Your selection can be  SA*ENA OR NB* OR *ASK’
Queue ‘You can further limit your selection by using Criteria’
Queue ‘Criteria can be any combination of’
Queue ‘  ID , CRE(ated Date in YY/MM/DD) And/Or CHA(nged Date in YY/MM/
DD’
Queue ‘Please ensure that the statistics have been saved in YY/MM/DD
format’
Queue ‘If not you may have to modify the zlcdate and zlmdate suitably’
Queue ‘Example Of Criteria:- ID=MOYEENKH&CHA<97/Ø1/Ø1’
Queue ‘                      ID<>HASLEFT’
Queue ‘                      CRE<97/Ø1/Ø1’
Queue "                      ID=’’ or ID<>’’"
Queue ‘<>=\^ are the only comparators that are permitted in Criteria.’
Queue ‘Use "&" to AND and "|" to OR the arguments'
Queue ‘To use the criteria you must pass it when you are calling'
xExecName
Queue ‘If you have not used criteria, then the program will ask you'
Queue ‘whether to delete members created/modified by others'
Do Until Queued()=Ø
 Parse Pull xHelp
 Say Copies(‘ ‘,5) xHelp
 End
Return
```

Moyeen Ahmed Khan
Canada

The FASTFIND accelerated string search

INTRODUCTION

Many applications perform a large amount of time searching for strings (searching for text, searching for table entries, etc). This subroutine (function) has been optimised to reduce the overhead of such searches.

ALGORITHM

A simple algorithm for searching for a phrase in a string could be performed as follows (in pseudo-code):

SET string index TO 1

DO loop FOREVER

IF phrase EQ string(index) THEN GOTO found

INCREMENT index BY 1

END loop

notfound:

This algorithm would typically require:

n compares (with the length of the phrase)

n increments to the index

n passes through the loop

Where n is either the character position in string where the phrase was found or the length of the string (phrase was not found).

The optimized search method uses the following algorithm:

SET string index TO 1

DO loop FOREVER

Search for the initial character of the phrase in the string(index)

IF NOT FOUND THEN GOTO notfound

SET index to the next found initial character

IF phrase EQ string(index) THEN GOTO found

INCREMENT index BY 1

END loop

This effectively reduces making the comparison for <phrase> only when the initial character of <phase> is found in <string>.

This algorithm would typically require:

m searches for the initial character of phrase

m compares (with the length of the phrase)

m increments to the index

m passes through the loop

Where m is the number of times the first character of the <phrase> occurs in the <string>.

A typical estimate for m could be approximately n/62. This assumes that <phase> and <string> consist of alphanumeric characters that occur with uniform frequency (62 = 26 uppercase characters + 26 lowercase characters + 10 numerics). This simple estimate shows that the optimised algorithm could generally be expected to yield a 62-fold (6200 percent) improvement over the standard search algorithm.

In practice the actual relative performance of the two algorithms depends on the data.

Best case for FASTFIND. <phrase> is not present in <string>. Only a single instruction is performed (search for the initial character), which yields the NOTFOUND condition. This corresponds to the worst case for the standard algorithm (the complete string must be

indexed through).

Worst case for FASTFIND. <string> contains only the initial character of <phrase>. This effectively reduces the two algorithms to the same, but with FASTFIND having the additional overhead of searching for the initial character. This is the only case where FASTFIND performs (a little) worse than the conventional search.

The performance of FINDFAST increases the more often the initial character of <phrase> is present in <string>.

IMPLEMENTATION

The TRT (Translate and Test) instruction is normally used to search for one or more characters (in this case the table would only contain the initial character of <phrase> as non-zero data). The use of TRT has two problems:

- The search is limited to 256 characters. TRT must be used repeatedly if <string> is longer.
- TRT cannot be used if the initial character of <phrase> is X'00'.

The new SRST (Search String) machine instruction avoids these problems.

Note: For performance reasons, <phrase> cannot be longer than 256 characters. This limitation permits the use of a simple CLC instruction. Obviously a CLCL could be used in its place (there would be additional overhead to set up the registers before each compare). There is no limitation on the length of <string>.

USE IN PRACTICE

Although the overhead for FASTFIND is not large (it mainly involves setting up the conditions for SRST) and it almost always performs better than the conventional search, there are some circumstances when its use is not appropriate:

- The searched string is short (say less than 20 characters)

- The search is performed infrequently.

On the other hand, there are circumstances where the use of FASTFIND is predestined:

- The searched string is large
- The initial character of search phrase occurs infrequently in the searched string

INVOCATION

CALL FASTFIND,(aphrase,lphrase,astring,lstring,ix)

<aphrase>: address of <phrase>

<lphrase>: length of <phrase> - binary word

<astring>: address to <string>

<lstring>: length of <string> - binary word

<ix>: index to the found phrase in <string> - binary word.

0 = not found

>0 = index in <string>. 1 = first position, etc.

The return value for FASTFIND (register 15) has the same content as <ix>.

SAMPLE USE

```
WORKING-STORAGE SECTION.
 01 PHR     PIC X(3) VALUE 'lai'.
 01 PHR-LEN PIC 9(8) BINARY VALUE 3.
 01 STR     PIC X(100) VALUE 'The rain in Spain lies mainly on the
plain'.
 01 STR-LEN PIC 9(8) BINARY VALUE 100.
 01 STR-IX  PIC S9(8) BINARY VALUE -1.

     CALL 'FASTFIND' USING PHR PHR-LEN STR STR-LEN STR-IX
      END-CALL
```

FASTFIND

```
         TITLE 'Function for Accelerated String Search'
* FASTFIND is a time-optimised function that
* searches for the specified phrase in a string.
* Invocation: CALL FASTFIND,(aphrase,lphrase,astring,lstring,ix)
* <aphrase>: address of <phrase>
* <lphrase>: length of <phrase> (fullword)
* <astring>: address to <string>
* <lstring>: length of <string> (fullword)
* <ix>: index of the found phrase (fullword). (0 = not found).
```

Problem diagnosis in MVS 'early'code

THE PROBLEM

Recently we encountered problems at subsystem initialization, which proved difficult to diagnose. The difficulty in diagnosing the problem was caused because when the MVS 'early' code abends then no system dump is taken. Instead control is passed to IEFJSBLD which produces a record in SYS1.LOGREC. The record in SYS1.LOGREC does not provide any clues as to why the system has terminated.

THE SOLUTION

We were provided with the following ZAP which causes MVS (IEFJSBLD) to take a system dump when an abend occurs during sub-system initialization.

```
++USERMOD(DB2ERLY)
++VER(Z038) FMID(HBB4430)
++ZAP(IEFJSBLD)
  NAME IEEVIPL IEFJSBLD
  VER 0CFE 47E0,CD1E
  REP 0CFE 4700,CD1E
```

Our problem turned out to be that we were using the wrong level early code, which the above ZAP enabled us to pinpoint immediately.

C Hunt
Systems Programmer (UK)

CONTROL CARD IMAGE QUICK BUILDER

Frequently within CLISTS and REXX EXECs, utilities are invoked that require input via control cards. While the control cards can be built within CLISTS and EXECs, it is not a user friendly process. For that reason, I came up with the CNTLCARD program, which makes it much easier to build the control card input required by such utilities.

The CNTLCARD program assumes (and requires) that a pre-allocated virtual I/O (VIO) dataset be used to hold the control card that is to be generated. This was done as a security feature, since sensitive information could be put into the control card (like passwords) and only the job or user allocating the VIO dataset could access it. It is also

the reason that only one control card can be written to the VIO dataset using the CNTLCARD program (you cannot specify a disposition of MOD for a VIO dataset). If you do not require the security, or if you desire to be able to create multiple control cards, you can modify the source code at label DDOK to skip the VIO checking. If you do not perform this modification, CNTLCARD will set a return code of 4 if the SYSIN DD does not refer to a VIO dataset.

CNTLCARD will return two other return codes. A code of 8 will be set if the SYSIN DD is not allocated, and a return code of 12 will be set if no parameter is passed to it. If you attempt to pass a control card whose length is greater than 80 bytes for CNTLCARD to write, it will truncate the length to 80 bytes without issuing any notification.

In order to handle special character situations, such as leading blanks, the string of data to be used as a control card can be enclosed in single quotes, which will be stripped off if found by the CNTLCARD program. Otherwise the string can be coded without the quotes even if it has embedded blanks or special characters.

As an example of its use, to generate a SORT control card, you would code:

```
SYSLOAD  ' SORT FIELDS=(3,9,CH,A),FILSZ=E100'
```

Below is an example CLIST which uses an unmodified CNTLCARD

program to write a single control card to a VIO dataset:

```
FREE DD(SYSIN) DELETE
ALLOC DD(SYSIN) UNIT(VIO)   SP(1) NEW REUSE +
      RECFM(F) LRECL(80) BLKSIZE(80) DSORG(PS)
SYSLOAD ' SORT FIELDS=(3,9,CH,A),FILSZ=E100'
```

Below is an example CLIST which to use a modified version of the CNTLCARD program (as described earlier) to write multiple control cards to a non-VIO dataset:

```
FREE DD(SYSIN) DELETE
ALLOC DD(SYSIN) UNIT(SYSDA) SP(1) MOD REUSE DA(TEMP) +
      RECFM(F) LRECL(80) BLKSIZE(80) DSORG(PS)
SYSLOAD ' SORT FIELDS=(3,9,CH,A),FIlSZ=E100'
SYSLOAD ' RECORD TYPE=F'
```

```
         TITLE 'CNTLCARD-COMMAND TO LOAD SYSIN FROM COMMAND PARM.'
CNTLCARD CSECT
         YREGS
         STM   R14,R12,12(R13)
         LR    R12,R15
         USING CNTLCARD,R12
         LA    R2,SAVE
         ST    R2,8(,R13)
         ST    R13,4(,R2)
         LR    R13,R2
         LR    R2,R1               SAVE CPPL POINTER.
         RDJFCB SYSIN              LOAD JFCB.
         LTR   R15,R15             DDNAME EXIST?
         BZ    DDOK
         LA    R15,8               LOAD ERROR CODE OF 8.
         B     EXIT                EXIT.
DDOK     TM    JFCBAREA+JFCFLGS1-JFCB,JFCVRDS VIO DATASET?
         BO    VIOOK
         LA    R15,4               LOAD ERROR CODE OF 4.
         B     EXIT                EXIT.
VIOOK    OPEN  (SYSIN,OUTPUT),TYPE=J   OPEN THE VIO DATASET.
         L     R3,0(,R2)           POINT TO PARM.
         LH    R5,0(,R3)           LOAD LENGTH.
         LH    R4,2(,R3)           LOAD OFFSET.
         LA    R4,4(,R4)           CREATE ...
         SR    R5,R4               ... LENGTH.
         LTR   R5,R5               ANYTHING THERE?
         BP    PARMOK              B IF YES.
         LA    R15,12              LOAD ERROR CODE.
         B     EXIT                EXIT.
```

```
PARMOK   CH    R5,=H'80'           GREATER THAN 80?
         BNH   PARMOKO             B IF NOT.
         LA    R5,80               TRUNCATE.
PARMOKO  LA    R3,0(R3,R4)         POINT TO PARM.
         CLI   0(R3),C''''         FIRST CHAR APOST?
         BNE   MPARM               B IF NOT.
         LA    R3,1(,R3)           BUMP POINTER
         SH    R5,=H'2'            ASSUME TRAILING APOST.
         BP    MPARM               B IF NOT MERELY ONE APOST.
         LA    R15,12              SET RC OF 12.
         B     EXIT                EXIT.
MPARM    BCTR  R5,*-*              GEN SS LEN.
         EX    R5,MOVEPARM         MOVE PARM.
         PUT   SYSIN,OUT           OUTPUT PARM.
         CLOSE (SYSIN)             CLOSE DATASET.
         SR    R15,R15             SET RC OF 0.
EXIT     L     R13,SAVE+4
         L     R14,12(,R13)
         LM    R0,R12,20(R13)
         BR    R14
MOVEPARM MVC   OUT(*-*),0(R3)      EXECUTED MOVE.
SAVE     DS    18F
EXLST    DC    X'87',AL3(JFCBAREA)
JFCBAREA DS    XL176
OUT      DC    CL80' '
         LTORG
         PRINT NOGEN
SYSIN    DCB   DDNAME=SYSIN,DSORG=PS,LRECL=80,BLKSIZE=(80),              *
               RECFM=F,BUFNO=1,NCP=1,MACRF=(PM),EXLST=EXLST
JFCB     DSECT
         IEFJFCBN LIST=NO
         END
```

MVS news

IBM has previewed Version 1 Release 5 of DFSMS/MVS, which promises much less overhead and redundancies than seen in earlier versions of HFS, with attendant performance gains. It will also support multi-volume datasets, provide an interface for carrying out additional functions such as retrieving statistics and dynamically extending HFS data sets, and preserving the integrity of mounts in a shared environment. Specifically, Version 1 Release 5 will improve the performance of shared catalog in a Parallel Sysplex environment, and provide the ability to maintain multiple HSMplexes within a single GRSplex. Each HSMsubplex will have its own set of control datasets and manage its own set of user catalogs.

The DFSMShsm control datasets will take advantage of VSAM extended addressing introduced in DFSMS/MVS 1.3. OAM will exploit sysplex architecture and DB2 data sharing, and will remove the 100 storage group limitation in a configuration. Users can now specify their own high-level qualifier for the object groups and their associated DB2 tables. OAM will also exploit sysplex architecture to enhance availability and allow instances of OAM which belong to an XCF group and are connected to DB2 subsystems, using DB2 data sharing for full access to OAM objects.

New functions include an API to allow for READ/WRITE access to the DFSMSrmm control dataset, allowing users to write programs to query and update CDS fields.

Contact your local IBM representative for further information.

* * *

Technology Investments subsidiary MessageQuest, an MQSeries specialist, is working with IBM to integrate IBM and Lotus products into its applications family. The first two products delivered under this effort are File Transfer Facility for MQSeries (FTF/MQ) and Configuration Management Tool for MQSeries (CMT/MQ).

FTF/MQ is said to provide an ability to exchange files between applications across the enterprise. By layering on top of MQSeries, FTF/MQ supports virtually all multi-platform, multi-protocol, and multi-operating system environments and delivery mechanisms.

CMT/MQ, meanwhile, is described as a full-featured MQSeries configuration and administration tool allowing real-time updates to MQSeries queue managers. It models MQSeries queue managers, and the channels that connect them, to form an enterprise-wide MQSeries network.

The software is based around three main components: a Java-based console, a central repository, and a distributed agent. The console provides a graphical presentation of queue managers and channels with point-and-click facilities for complex operations such as creating new channels between queue managers.

CMT/MQ and FTF/MQ are both available, and support MVS/ESA, NT, major Unix systems, and OS/400.

Contact your local IBM representative for further information.

* * *