In this issue

3  Expanded storage activity monitor
7  Unreferenced interval count distribution
16 Synchronizing remote PDS members
33 WLM in an sysplex environment
58 Year 2000 aid: replace source strings
72 MVS news

© Xephon plc 1998
Editor
Dr Jaime Kaminski

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

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

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

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

Printed in England.
INTRODUCTION

Expanded storage (ESTOR) is a key performance area of IBM mainframes today. It is used by many system software components in a variety of ways to boost the overall amount of useful work a complex can perform by significantly reducing the time required to do that work.

An adage has arisen that ‘the fastest I/O is the I/O that you do not have to perform’. This refers to the ability of products such as database management systems to maintain large buffers of user data in ESTOR areas called dataspaces, rather than having to rely on accessing this data from relatively slow devices such as DASD.

While the use of ESTOR has grown enormously since its introduction around 10 years ago, and with it the importance of this resource to the enterprise, tools for monitoring ESTOR activity have not mirrored this growth.

As is often the case, MVS records a certain amount of interesting information in common control blocks which are easily accessed with the REXX ‘storage’ function. In this case, the control block of interest is the Real Storage Manager Control and Enumeration Area (RCE), which is pointed to by the Communications Vector Table (CVT).

The RCE contains a wide range of information including such things as how many storage frames, both central and expanded, are on-line and available to the system, and counts of various types of pages paged in and out to (and from) ESTOR and auxiliary storage. I wrote a REXX routine, called ESTRMON, which picks up some of the counters from the RCE and displays them on an ISPF panel, ESTRMONP, which must be in the ISPF session panel library concatenation. Another panel, ESTRMONH, is invoked from ESTRMONP by pressing PFI and serves to explain the function of the program and the abbreviations used.

The program reads the RCE to initialize a set of variables, then loops
around reading new values of these variables and calculating the difference since the previous read. When divided by an elapsed time factor, this gives the rate at which the various events are occurring, and it is this information that is displayed on the panel. Each loop is triggered by the user pressing ‘enter’ at the terminal, and the process continues until the user presses PF3.

A useful feature is the ‘Refresh’ option, which enables the user to choose whether to use the initial starting time and values as a base for the rate calculations, or to refresh the base values every time. This means that the displayed counts and rates can be either since the session started (i.e. since ESTRMON was invoked) or since the previous pressing of ‘enter’.

The subset of RCE information chosen for this display was to suit the installation where I developed the program; other sites may wish to modify the fields displayed slightly as per their own interests and requirements. Fortunately this is generally extremely simple as many of the RCE counters are of the same 4-byte signed format as the fields that I have displayed. Only the hex offsets in the initialization and main loop sections of the program need be modified as per the offsets recorded in the RCE description in the Data Areas manuals, as well as the field description in the ISPF panel. For convenience, I have included the RCE field name suffix for the fields I used in a comment in the initialization section.

ESTRMON

```rexx
/*------------------------------ REXX -------------------------------*/
/* Function   : Expanded storage activity monitor                      */
/*------------------------------ */
/* Initialization                                              */
/*------------------------------ */
numeric digits 21
ref = 'Y'
cvt = storage(d2x(16),4)
rce = storage(d2x(c2d(cvt)+c2d(x2c(Ø49Ø))),4)) /* eswrt */
o.1 = c2d(storage(d2x(c2d(c2d(rce)+c2d(x2c(Ø0a8))),4))) /* eswrt */
o.2 = c2d(storage(d2x(c2d(rce)+c2d(x2c(Ø0ac))),4))) /* esrea */
o.3 = c2d(storage(d2x(c2d(rce)+c2d(x2c(Ø0cc))),4))) /* nwsf */
o.4 = c2d(storage(d2x(c2d(rce)+c2d(x2c(Ø0d4))),4))) /* wsdne */
o.5 = c2d(storage(d2x(c2d(rce)+c2d(x2c(Ø100))),4))) /* hspew */
o.6 = c2d(storage(d2x(c2d(rce)+c2d(x2c(Ø104))),4))) /* hsper */
```
o.7 = c2d(storage(d2x(c2d(rce)+c2d(x2c(Ø1Ø8))))),4)) /* hspe */
o.8 = c2d(storage(d2x(c2d(rce)+c2d(x2c(Ø1Øc))))),4)) /* hspo */
o.9 = c2d(storage(d2x(c2d(rce)+c2d(x2c(Ø11Ø))))),4)) /* hsppi */
o.10 = c2d(storage(d2x(c2d(rce)+c2d(x2c(Ø124))))),4)) /* bppie */
o.11 = c2d(storage(d2x(c2d(rce)+c2d(x2c(Ø128))))),4)) /* bppia */
o.12 = c2d(storage(d2x(c2d(rce)+c2d(x2c(Ø12c))))),4)) /* bpste */
o.13 = c2d(storage(d2x(c2d(rce)+c2d(x2c(Ø13Ø))))),4)) /* bpsta */
o.14 = c2d(storage(d2x(c2d(rce)+c2d(x2c(Ø134))))),4)) /* blpie */
o.15 = c2d(storage(d2x(c2d(rce)+c2d(x2c(Ø138))))),4)) /* blpia */
o.16 = c2d(storage(d2x(c2d(rce)+c2d(x2c(Ø13c))))),4)) /* bpste */
o.17 = c2d(storage(d2x(c2d(rce)+c2d(x2c(Ø14Ø))))),4)) /* blsta */
o.18 = c2d(storage(d2x(c2d(rce)+c2d(x2c(Ø144))))),4)) /* espi */
o.19 = c2d(storage(d2x(c2d(rce)+c2d(x2c(Ø148))))),4)) /* esst */
elap = time('E')
/*--------------------------*/
/* Main loop */
/*--------------------------*/
do forever
  aec = c2d(storage(d2x(c2d(rce)+c2d(x2c(Ø̄0⁹4))))),4))
  espl = c2d(storage(d2x(c2d(rce)+c2d(x2c(Ø̄0a0))))),4))
  esmb = format(espl/256,4,0)
  esinu = c2d(storage(d2x(c2d(rce)+c2d(x2c(Ø̄0a⁴))))),4))
  n.1 = c2d(storage(d2x(c2d(rce)+c2d(x2c(Ø̄0a⁸))))),4))
  n.2 = c2d(storage(d2x(c2d(rce)+c2d(x2c(Ø̄0ac))))),4))
  n.3 = c2d(storage(d2x(c2d(rce)+c2d(x2c(Ø̄0cc))))),4))
  n.4 = c2d(storage(d2x(c2d(rce)+c2d(x2c(Ø̄0d4))))),4))
  n.5 = c2d(storage(d2x(c2d(rce)+c2d(x2c(Ø̄100))))),4))
  n.6 = c2d(storage(d2x(c2d(rce)+c2d(x2c(Ø̄104))))),4))
  n.7 = c2d(storage(d2x(c2d(rce)+c2d(x2c(Ø̄108))))),4))
  n.8 = c2d(storage(d2x(c2d(rce)+c2d(x2c(Ø̄11c))))),4))
  n.9 = c2d(storage(d2x(c2d(rce)+c2d(x2c(Ø̄110))))),4))
  n.10 = c2d(storage(d2x(c2d(rce)+c2d(x2c(Ø̄124))))),4))
  n.11 = c2d(storage(d2x(c2d(rce)+c2d(x2c(Ø̄128))))),4))
  n.12 = c2d(storage(d2x(c2d(rce)+c2d(x2c(Ø̄12c))))),4))
  n.13 = c2d(storage(d2x(c2d(rce)+c2d(x2c(Ø̄130))))),4))
  n.14 = c2d(storage(d2x(c2d(rce)+c2d(x2c(Ø̄134))))),4))
  n.15 = c2d(storage(d2x(c2d(rce)+c2d(x2c(Ø̄138))))),4))
  n.16 = c2d(storage(d2x(c2d(rce)+c2d(x2c(Ø̄13c))))),4))
  n.17 = c2d(storage(d2x(c2d(rce)+c2d(x2c(Ø̄140))))),4))
  n.18 = c2d(storage(d2x(c2d(rce)+c2d(x2c(Ø̄144))))),4))
  n.19 = c2d(storage(d2x(c2d(rce)+c2d(x2c(Ø̄148))))),4))
  do i = 1 to 19
    e.i = n.i - o.i
  end
  if ref = 'N' then; do
    elap = time('E')
  end; else; do
    elap = time('R')
  do i = 1 to 19
    o.i = n.i
  end

do i = 1 to 19
    r.i = format(e.i/elap,5,1)
end
address ispexec "display panel(ESTRMONP)"
if rc ≠ Ø then
    do; exit Ø; end
end
exit

ESTRMONH PANEL
)
attr
    $ type(output) intens(high) just(right)
)body expand(@@)
%@@-@  Expanded Storage Activity Monitor  @-@
%COMMAND  ===>_ZCMD
+  +  A display of expanded storage activity counts and rates using
+  +  data extracted from the RCE. Abbreviations as follows:
+  +  ES = Expanded storage
+  +  Fr = Frames
+  +  ws = working set
+  +  HS = Hiperspace
+  +  aux = auxiliary storage
+  +  pg = paged
+  +  BP = Blocked pages
+  +  BL = Blocks of pages
+  +  Pgs = Pages
+  +  fited = Faulted in
+
)init
)proc
&zcont = ESTRMONH
)end

ESTRMONP PANEL
)
attr
    ! type(text) color(green)
    #$ type(output) color(yellow) just(right)
    ~ type(output) color(yellow) just(left)
    $ type(output) intens(high)
    " type(text) color(turq)
    type(text) skip(on) intens(low)
)body expand(@@)
%@@-@  Expanded Storage Activity Monitor  @-@
Unreferenced interval count distribution

INTRODUCTION

The tools available for tracking how efficiently a system’s central storage (CSTOR) is being used have always seemed to me to be a little thin on the ground. Unreferenced Interval Count (UIC) for the system being a very low value is seen as undesirable, and we know that the goals of the System Resources Manager (SRM) includes avoiding this situation where possible, but otherwise what can we really find out?
I have always been of the opinion that the more information one can dig out of the operating system the better. For capacity planners and performance analysts, the need to know is somewhat greater than the desire to inform demonstrated by much of MVS and its accompanying subsystems. Hence the profusion of third-party monitoring products, but, in the absence of such, it is simply amazing how much is available if you have the time and inclination to browse through the *Data Areas* manuals and put some code together.

I decided that I wanted to know more about the UIC distribution, not just the system high value. This would give me a better insight into how much of the CSTOR was really heavily accessed, and how much was living a relatively sedentary life.

**UNREFERENCED INTERVAL COUNT**

Perhaps a word or two about UIC would be in order. The Real Storage Manager (RSM) maintains a structure with a control block for each frame of central storage in the system known as the Page Frame Table (PFT). Each entry (PFTE) is 32 bytes and contains such information as which queue a queued frame is on and which address space is holding a frame. It also has a byte known as the UIC for the frame – this is an indicator of how long has elapsed since the frame was last referenced by the address space or common area it is associated with. The value is updated periodically by the SRM or the RSM and it ranges from 0 to 255.

SRM uses, and RMF reports on, the system-wide high UIC value in a number of decisions relating to adjusting workload to ease CSTOR constraint. The idea behind this is to check how long ago each frame in the system was referenced. If there is a frame that has not been referenced during the last 255 seconds or more (the unit of UIC is really seconds, each ‘interval counted’ is one second in duration, but because it is only 1 byte, it cannot exceed 255 in value) then the system high UIC is also 255. As this system high-value decreases, the system is said to be becoming more storage constrained and eventually SRM will endeavour to swap out jobs to relieve the situation.

All well and good, but as I pointed out, the only value that you can easily get hold of is the system high UIC. I was interested in how much
of the CSTOR was sitting at or near this value, and how much was at much lower UIC values.

The access to this information is through the PFTEs, and using the REXX ‘storage’ function I was able to run through the PFT and calculate the distribution. This was however a little heavy on the system, especially after an upgrade to 2GB of CSTOR, so I decided to recode the core access code in an Assembler routine which made the process much faster and more efficient.

A REXX program UICDISX is still used to drive the process and to handle the ISPF table and screen formatting; I find REXX admirably suited to such tasks. The Assembler routine UICDISA runs the PFTE chain and uses registers 2 through 9 as counters for various cut-off values of UIC. These values were somewhat arbitrarily chosen by me at UIC = 0, less than 10, less than 20, etc. If you require different cut-off values, the changes to be made are only to the lines of UICDISA such as:

```
@VLT10 CLC UICVALUE,=XL1'09'
```

as appropriate, as well as the line:

```
opt = ' = 0 < 10 < 20 < 40 < 80 < 160 < 255  Max'
```

in UICDISX to match your values. The expansions for the PFTE, RIT, and PVT may not be delivered in all versions of MVS, but to code your own macros using the Data Areas manuals as a guide is very easy.

UICDISX SOURCE

```
/*------------------------------ REXX -------------------------------*/
/* Function   : UIC distribution display                             */
/*-------------------------------------------------------------------*/
numeric digits 15
opt = ' = 0 < 10 < 20 < 40 < 80 < 160 < 255  Max'
do forever
  tfr = 0
  "CALL 'USER.LINKLIB(UICDISA)'"
  nl = ' rng uic prc bar'
  address ispexec "tbcreate uictab names("nl") nowrite replace"
  do i = 1 to 8
    j = 'X' || right(d2x(i),8,0)
    uic = c2d(uc.j)
    tfr = tfr + uic
```

```plaintext
do i = 1 to 8
  j = 'X' || right(d2x(i),8,Ø)
  uic = c2d(uc.j)
  k = i * 5 - 4
  rng = substr(opt.k,5)
  prc = format(uic * 100 / tfr,3,1)
  bgg = '....+....+....+....+....+....+....+....+....+....+'
  bar = overlay(beg,bcg,5,Ø)
  address ispexec "tbadd uictab"
end

rng = ''; uic = ''; prc = ''; bar = ''
address ispexec "tbadd uictab"
rng = 'Total'
uic = tfr
address ispexec "tbadd uictab"
address ispexec "tbtop uictab"
address ispexec "tbdispl uictab panel(UICDISP)"
if rc ≠ Ø then
do
  address ispexec "tbclose uictab"
  exit
end
address ispexec "tbclose uictab"
end
exit

UICDISP PANEL

)attr
! type(output) color(yellow) just(right)
~ type(output) color(red) just(left)
$ type(output) color(green) just(right)
# type(output) color(yellow)
  type(text) skip(on) intens(low)
)body expand(@@)
%@-@ Unreferenced Interval Count Distribution @-@
%COMMAND  ===>_ZCMD                                     +     %SCROLL ===>_AMT
+
%
% UIC  Frames  Perc  10  20  30  40  50  60  70  80  90  100
%
)model
```
%COMMAND  ===>_ZCMD                             +     %SCROLL ===>_AMT +
%
%      UIC measures how long ago a frame of CSTOR was referenced by the
%      system. This function shows the distribution of UIC values for
%      all the frames on the system.
%
%      UICDISA CSECT
STM  14,12,12(13)          SAVE CALLER'S REGISTERS
USING UICDISA,12            ESTABLISH ADDRESSABILITY

LR  12,15                SET UP MY BASE  
LR  14,13                SAVE ADDR(PREVIOUS SAVE AREA)  
LA 13,SAVE               ADDR(MY SAVE AREA)  
ST 13,8(.14)             CHAIN MY SAVE AREA TO PREVIOUS  
ST 14,4(.13)             CHAIN PREVIOUS SAVE AREA TO MINE  
LM 14,1,12(13)           RESTORE REGS 14 ---> 1  

@CVTPTR  L 11,CVTPTR  
USING CVT,11  
L 11,CVTPVTP  
DROP 11  
USING PVT,11  
L 11,PVTRIT  
DROP 11  
USING RIT,11  
L 10,RITPFTEN  
L 11,RITPFTE1  
DROP 11  
USING PFTE,11                ADDR FIRST PFTE  
XR 9,9                 UIC = Ø  
XR 8,8                 UIC L 10  
XR 7,7                 UIC L 20  
XR 6,6                 UIC L 40  
XR 5,5                 UIC L 80  
XR 4,4                 UIC L 160  
XR 3,3                 UIC L 255  
XR 2,2                 UIC = 255  

@PROCPFT  MVC UICVALUE,PFTEUIC                GET UIC FROM PFTE  
CLC UICVALUE,=XL1'ØØ'                IS UIC = Ø  
BNE @VLT1Ø  
LA 9,1(.9)  
B @NEXT  

@VLT1Ø  CLC UICVALUE,=XL1'Ø9'  
BH @VLT2Ø  
LA 8,1(.8)  
B @NEXT  

@VLT2Ø  CLC UICVALUE,=XL1'13'  
BH @VLT4Ø  
LA 7,1(.7)  
B @NEXT  

@VLT4Ø  CLC UICVALUE,=XL1'27'  
BH @VLT8Ø  
LA 6,1(.6)  
B @NEXT  

@VLT8Ø  CLC UICVALUE,=XL1'4F'  
BH @VLT16Ø  
LA 5,1(.5)  
B @NEXT  

@VLT16Ø  CLC UICVALUE,=XL1'9F'  
BH @VLT255  
LA 4,1(.4)
B @NEXT
@VLT255 CLC UICVALUE.=XL1'FF'
BNE @VEQ255
LA 3,1(,3)
B @NEXT
@VEQ255 LA 2,1(,2)
*
@NEXT CR 11,10 IS CURRENT PFTE LAST PFTE ?
BE @UPDREX NO
LA 11,32(,11) PROCESS NEXT PFTE
B @PROCPT LOOP
*
** UPDATE REXX VARIABLES
*
@UPDREX EQU *
ST 2,CMAX
ST 3,C255
ST 4,C160
ST 5,C080
ST 6,C040
ST 7,C020
ST 8,C010
ST 9,C000
XR 6,6 ZERO REG 6
MVC NAME.=CL4'UC.X'
MVC NL.=F'12'
MVC VL.=F'4'
LA 6,1(,6) INCREMENT BY 1
REGDISP 6,NAMIDX
LA 8,C000
ST 8,VP
LINK EP=IKJCT441,PARAM=(ECU,NP,NL,VP,VL,TK),VL=1
LA 6,1(,6) INCREMENT BY 1
REGDISP 6,NAMIDX
LA 8,C010
ST 8,VP
LINK EP=IKJCT441,PARAM=(ECU,NP,NL,VP,VL,TK),VL=1
LA 6,1(,6) INCREMENT BY 1
REGDISP 6,NAMIDX
LA 8,C020
ST 8,VP
LINK EP=IKJCT441,PARAM=(ECU,NP,NL,VP,VL,TK),VL=1
LA 6,1(,6) INCREMENT BY 1
REGDISP 6,NAMIDX
LA 8,C040
ST 8,VP
LINK EP=IKJCT441,PARAM=(ECU,NP,NL,VP,VL,TK),VL=1
LA 6,1(,6) INCREMENT BY 1
REGDISP 6,NAMIDX
LA 8,C080

REGDISP MACRO

*******************************************************************************
** Convert the contents of a passed register to an 8-character display field. **
*******************************************************************************
MACRO
&LABEL REGDISP &HEX,&DSP
&LABEL STM Ø,15,SAVE&SYSNDX
ST &HEX,WHEX&SYSNDX
UNPK WDSP&SYSNDX.(9),WHEX&SYSNDX.(5)
NC WDSP&SYSNDX.(8),MASK&SYSNDX
TR WDSP&SYSNDX.(8),HXTB&SYSNDX
MVC &DSP,WDSP&SYSNDX
LM Ø,15,SAVE&SYSNDX
B END&SYSNDX
SAVE&SYSNDX DS 16F
MASK&SYSNDX DC XL8'ØFØFØFØFØFØFØFØF'
HXTB&SYSNDX DC CL16'0123456789ABCDEF'
WHEX&SYSNDX DS F
DS C
WDSP&SYSNDX DS CL8'********'
DC CL1'.'
END&SYSNDX DS ØH
MEND

Patrick Mullen
MVS Systems Consultant (Canada) © Xephon 1998
Synchronizing remote PDS members

Many systems programmers and other IT staff need to maintain the contents of partitioned datasets over multiple systems. Sometimes shared DASD makes things easier, but with remote sites it is difficult to keep members of similar libraries (for example PROCLIBs and PARMLIBs) in step. Bulk file transfer is one method, but it’s rather crude to copy all the members and time-consuming to select members manually.

This utility uses several interesting techniques to allow just the PDS members that have changed to be selectively and automatically transferred on request. Typically the entire dataset would be copied initially and then this utility used to maintain the PDS in-line across the remote systems. It detects changes using the ISPF member statistics and transfers members using a TSO XMIT command in a special way that does not require manual invention to receive it.

```
- - - - - - - - - - - - - - - - - - - P D S  S Y N C H R O N I Z A T I O N - - - - - - - - - - - - - -

OPTION ===>

1  TRANSMIT - Send changed members since last snapshot and take new snapshot
2  NEW SNAP - Record a snapshot of member statistics (replace existing snap)
3  OLD SNAP - Rename previous snapshot to recorded snapshot (if xmit failed)
4  DIS SNAP - Display recorded snapshot member statistics (if any)
5  DEL SNAP - Delete the recorded snapshot member statistics (if any)

NOTE : Only members with ISPF (or PDSMAN) statistics will be processed.
Specify 'DATASET' in TSO syntax (do not specify a member name)
DATASET NAME ===> 'SYS1.TEST.PDS'      (On both systems)
Review xmit ===> Y    (Y/N, to review changed list before transmit)
To MVS node ===> ANOTHER    (second job will execute here)

Batch JOBS (jobname XR44X/R) First job generates xmit, second receives it
JOB Acct ===> ABC112
JOB Class ===> A    JOB msgclass ===> X

Enter details and press ENTER to continue or press END to exit
```

Figure 1: Panel displayed after SYNC 1 is invoked
To install this facility, copy the REXX to a SYSPROC library, the panels to an ISPLLIB library, the skeleton to an ISPSLIB, the message member to an ISPMLIB library, and finally assemble and link the Assembler program (this is optional for initial testing purposes). Invoke the SYNC1 REXX under ISPF. A panel is displayed like the one shown in Figure 1.

If the ‘review xmit’ option is selected, then a member list of changed members is shown. These are about to be file transferred and will be unless the command ‘CANCEL’ is entered in the command field. Changes are members where the date or time of last modification has altered since the last snapshot was taken, according to ISPF statistics. The ISPF option 3.5 can be used to force an update to these statistics.

If all is well then ENTER is pressed and JCL is generated for the first batch job on the local node. This job prepares the file transfer JCL and then submits it. The first job’s JCL looks like this:

```
//XR44XF JOB ABC112,'XR44',
//  NOTIFY=XR44,
//  CLASS=A,MSGCLASS=X
//*
//** GENERATE FILE TRANSFER ON LOCAL FOR REMOTE SUBMIT
//**
//STEPX EXEC PGM=IKJEFT01,DYNAMNBR=90,REGION=4M
//SYSPROC DD DSN=SYS1.REXX.LIB,DISP=SHR
//SYSTSPRT DD SYSOUT=*```
The SYNC2 REXX (which also runs on the origin node) will read the list of members and generate the job that file transfers the members. This REXX can also used as a stand-alone file transfer facility by using JCL similar to the above example. The generated JCL from SYNC2 will contain the member data as well as the remote invocation of SYNC3, which issues the RECEIVE command for the members at the remote end.

It is important to understand that the XMIT data travels with the JCL to the remote node, unlike using XMIT in the conventional way. This is achieved by redirecting the data output from the XMIT command and the same is performed by the RECEIVE command at the other end.

If desired, the stored snapshot can be examined and the ISPF display will look like the one in Figure 3. This shows the member’s status at the time of the last snapshot or transfer point. Note that member content is not merged on a line-by-line basis, the entire member is replaced.

Several options are available. Normally one starts tracking changes by taking a ‘snapshot’. This stores the current ISPF statistics in a special member (###SNPCR). Subsequent changes to members can then be detected and used to initiate a selective transfer (option 1).

The transfer option submits a job for local execution that then generates the file transfer job by generating instream XMIT data inside a RECEIVE job, which in turn executes on the remote node. The snapshot member is also updated to the current ISPF statistics.
In the event of a file transfer failure, a back-up (old) copy of the snapshot can be restored to allow another attempt. Other options allow the deletion of the snapshot or the display of its contents. Members without ISPF statistics are not processed during transfer. The back-up snapshot is stored in a member with the name of ###SNPOD.

This utility is not intended for massive volumes of data and other programs such as FTP or Netview File Transfer should be used for the initial bulk copying. But it will allow convenient updating of libraries that normally should have the same content. It can work in either/both directions and does not need to make one system the master end. The list of members about to be transferred can be shown before the job is submitted and a cancel option is available.

An Assembler program is provided to issue ENQs and DEQs. This is used at two points – once to ensure that no-one else is trying to synchronize the same library at exactly the same time, and also to update the target library without using DISP=OLD. This uses the standard shared-write technique of ISPF, the SPFEDIT ENQ. During

<table>
<thead>
<tr>
<th>MEMBER</th>
<th>DATE</th>
<th>TIME</th>
<th>USER</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABC</td>
<td>92/03/16</td>
<td>16:35</td>
<td>XR44</td>
</tr>
<tr>
<td>CCC1</td>
<td>95/08/04</td>
<td>09:40</td>
<td>DFDT</td>
</tr>
<tr>
<td>CMF</td>
<td>96/08/25</td>
<td>13:49</td>
<td>DFDT</td>
</tr>
<tr>
<td>COLCL</td>
<td>88/06/01</td>
<td>10:05</td>
<td>SSPR</td>
</tr>
<tr>
<td>COMP</td>
<td>97/07/19</td>
<td>14:25</td>
<td>SSPR</td>
</tr>
<tr>
<td>COPY2</td>
<td>96/05/11</td>
<td>21:37</td>
<td>LARS</td>
</tr>
<tr>
<td>COUNT</td>
<td>95/06/01</td>
<td>10:06</td>
<td>SSPR</td>
</tr>
<tr>
<td>CPYBATCH</td>
<td>93/06/11</td>
<td>06:52</td>
<td>LARS</td>
</tr>
<tr>
<td>CSPC1</td>
<td>91/08/13</td>
<td>09:36</td>
<td>ERER</td>
</tr>
<tr>
<td>TEST1</td>
<td>97/11/14</td>
<td>16:38</td>
<td>XR44</td>
</tr>
<tr>
<td>TEST2</td>
<td>97/11/14</td>
<td>16:34</td>
<td>XR44</td>
</tr>
<tr>
<td>TEST3</td>
<td>97/11/14</td>
<td>16:38</td>
<td>XR44</td>
</tr>
</tbody>
</table>

Figure 3: ISPF display
initial testing, you can leave out this program, but it should be in before production use.

The SYNC1 REXX program demonstrates a useful interface to ISPF that handles member statistics (the LMMLIST call). It also shows how to process a data-table within an ISPF skeleton. SYNC2 REXX shows how to generate instream XMIT jobs, and SYNC3 how to receive them. The programs are fully working as supplied and the panels prompt for installation-specific parameters such as account code, msgclass, etc, but please test the utility carefully before making it available. Also, change the references to SYS1.REXX.LIB to your own SYSPROC library.

SYNC1
/* REXX - SYNC1 - PDS SYNCHRONIZATION MEMBER UPDATE TRANSFER    */
/* RUNS ON LOCAL NODE IN FOREGROUND UNDER ISPF                */
/*                                                                  */
/* PART 1, SNAPSHOT CREATE, TRANSMIT SUBMIT, OLD SNAPSHOT ETC     */
/*******************************************************************/
CALL INITIAL
ADDRESS ISPEXEC "DISPLAY PANEL(SYNCP1)"
PRC = RC
DO WHILE (PRC = Ø)
   CALL PROCESS ZCMD
   ADDRESS ISPEXEC "DISPLAY PANEL(SYNCP1)"
   PRC = RC
END

CALL TERMIN
EXIT Ø
/* INITIAL, SET UP VARIABLES FOR HOST AND TARGET SYSTEM          */

INITIAL:
ADDRESS ISPEXEC "VGET (ZSCREEN) SHARED"
USER = USERID()
SNAP = '###SNPCR' /* CONTROL MEMBER NAME */
SNAPO = '###SNPOD' /* OLD VERSION OF IT */

ADDRESS ISPEXEC "VGET (SDSN SLST STOS SECL SMCL SACT) ASIS"
IF SACT = ' ' THEN /* FIND FROM RACF LU */
   CALL GET_LOGON_ACCT
IF SLST = '' THEN
   SLST = 'Y'
RETURN Ø
/* TERMIN, SET UP VARIABLES FOR TERMINATION */
TERMIN:
ADDRESS ISPEXEC "VPUT (SDSN SLST STOS SECL SMCL SACT) PROFILE"
RETURN Ø

/* PROCESS, ACCEPT DSN, OPTION TO CHECKPOINT OR GENERATE MEMLIST */
PROCESS:
ARG OPTION
IF SUBSTR(SDSN,1,1) ≠ '¨' THEN
  SDSN = '¨USER'.'SDSN'¨'
IF RIGHT(SDSN,1) ≠ '¨' THEN
  SDSN = SDSN'¨'
STATUS = SYSDSN(SDSN)
IF STATUS ≠ 'OK' THEN
  DO
    ADDRESS ISPEXEC "SETMSG MSG(SYNC001)"
    RETURN 4
  END
X = LISTDSI(SDSN)
IF SYSREASON > Ø THEN
  DO
    ADDRESS ISPEXEC "SETMSG MSG(SYNC004)"
    RETURN 4
  END
DSORG = SYSDSORG
RECFM = SYSRECFM
LRECL = SYSLRECL
IF DSORG ≠ 'PO' | RECFM = 'U' THEN
  DO
    ADDRESS ISPEXEC "SETMSG MSG(SYNC003)"
    RETURN 4
  END
CALL SERIAL 'ENQ'
IF OPTION = '1' THEN
  CALL TRANSMIT
IF OPTION = '2' THEN
  CALL SNAPSHOT
IF OPTION = '3' THEN
  CALL OLDSNAP
IF OPTION = '5' THEN
  CALL DELSNAP
CALL SERIAL 'DEQ'
IF OPTION = '4' THEN
  CALL RECSNAP
RETURN Ø

/* SNAPSHOT, GENERATE CURRENT MEMBER LIST AND SAVE AS SNAP */
SNAPSHOT:
CALL CURLIST
IF RESULT > Ø THEN
  RETURN 4
CALL SETSNAP
IF RESULT = Ø THEN
   ADDRESS ISPEXEC "SETMSG MSG(SYNCØØ6)"
RETURN Ø
/* TRANSMIT, GENERATE FILE TRANSFER JOB OF NEW/CHANGED MEMBERS */
TRANSMIT:
CALL OLDLIST
IF RESULT > Ø THEN
   RETURN 4
CALL CURLIST
IF RESULT > Ø THEN
   RETURN 4
CALL COMPARE
IF RESULT > Ø THEN
   RETURN 4
CALL SUBMIT
IF RESULT > Ø THEN
   RETURN 4
CALL SETSNAP
RETURN Ø
/* OLDSNAP, COPY PREVIOUS SNAP MEMBER TO CURRENT ONE */
OLDSNAP:
CALL RENAME SNAPO SNAP
IF RESULT > Ø THEN
   DO
      ADDRESS ISPEXEC "SETMSG MSG(ZERRMSG)"
      RETURN 4
   END
IF RESULT = Ø THEN
   ADDRESS ISPEXEC "SETMSG MSG(SYNCØØ7)"
RETURN Ø
/* RECSNAP, DISPLAY RECORDED SNAP LIST AS TABLE */
RECSNAP:
CALL CURLIST
IF RESULT > Ø THEN
   RETURN 4
CALL OLDLIST
IF RESULT > Ø THEN
   RETURN 4
TABLE = 'SYNC' || ZSCREEN
ADDRESS ISPEXEC "TBCREATE" TABLE "NAMES(INFO) NOWRITE REPLACE"
DO I = 1 TO OLDSTAT.Ø
   INFO = OLDSTAT.I
   ADDRESS ISPEXEC "TBADD" TABLE "MULT(5)"
END
ADDRESS ISPEXEC "TBTOP" TABLE
ADDRESS ISPEXEC "TBDISPL" TABLE "PANEL(SYNCP4)"
ADDRESS ISPEXEC "TBEND" TABLE
RETURN Ø
/* DELSNAP, DELETE RECORDED MEMBER LIST */
DELSNAP:
ADDRESS ISPEXEC "DISPLAY PANEL(SYNCP5)"
IF RC > Ø THEN
  RETURN 4
ADDRESS ISPEXEC "LMINIT DATAID(DID) DATASET("SDSN") ENQ(SHRW)"
IF RC > Ø THEN
  DO
    ADDRESS ISPEXEC "SETMSG MSG("ZERRMSG")"
    RETURN 4
  END
ADDRESS ISPEXEC "LMOPEN DATAID("DID") OPTION(OUTPUT)"
IF RC > Ø THEN
  DO
    ADDRESS ISPEXEC "SETMSG MSG("ZERRMSG")"
    ADDRESS ISPEXEC "LMFREE DATAID("DID")"
    RETURN 4
  END
ADDRESS ISPEXEC "LMFREE DATAID("DID")"
ADDRESS ISPEXEC "LMCLOSE DATAID("DID")"
MRC = RC
ADDRESS ISPEXEC "LMOPEN DATAID("DID")"
ADDRESS ISPEXEC "LMFREE DATAID("DID")"
IF MRC = Ø THEN
  ADDRESS ISPEXEC "SETMSG MSG(SYNCO12)"
ELSE
  ADDRESS ISPEXEC "SETMSG MSG(SYNCO13)"
END
RETURN Ø
/* CURLIST, GENERATE CURRENT MEMBER LIST FROM PDS STATISTICS */
CURLIST:
  I = Ø
  MEM = ' '
  SNAPDATE = ' '
  SNAPTIME = ' '
ADDRESS ISPEXEC "LMINIT DATAID(DID) DATASET("SDSN") ENQ(SHR)"
IF RC > Ø THEN
  DO
    ADDRESS ISPEXEC "SETMSG MSG("ZERRMSG")"
    RETURN 4
  END
ADDRESS ISPEXEC "LMOPEN DATAID("DID") OPTION(INPUT)"
IF RC > Ø THEN
  DO
    ADDRESS ISPEXEC "SETMSG MSG("ZERRMSG")"
    ADDRESS ISPEXEC "LMFREE DATAID("DID")"
    RETURN 4
  END
ADDRESS ISPEXEC "LMMLIST DATAID("DID") OPTION(LIST), "STATS(YES) MEMBER(MEM)"
MLC = RC
DO WHILE (MLC = Ø)
  IF SUBSTR(MEM,1,3) ≠ '###' & ZLMDATE ≠ ' ' THEN
DO
    I = I + 1
    CURSTAT.I = MEM ZLMDATE ZLMTIME ZLUSER
END
IF MEM = SNAP THEN
    DO
        SNAPDATE = ZLMDATE
        SNAPTIME = ZLMTIME
    END
ADDRESS ISPEXEC "LMMLIST DATAID("DID") OPTION(LIST)",
    "STATS(YES) MEMBER(MEM)"
    MLC = RC
END
CURSTAT.Ø = I
ADDRESS ISPEXEC "LMMLIST DATAID("DID") OPTION(FREE)"
ADDRESS ISPEXEC "LMCLOSE DATAID("DID")"
ADDRESS ISPEXEC "LMFREE  DATAID("DID")"
RETURN Ø

/* SETSNAP, SAVE CURRENT MEMBER LIST AS MEMBER SNAP            */
SETSNAP:
CALL RENAME SNAP SNAPO
ADDRESS ISPEXEC "VGET (ZDATE ZTIME) SHARED"
DROP ZLVERS ZLMOD ZCNORC ZLINORC ZLMNORC
ZLCDATE = ZDATE
ZLMDATE = ZDATE
ZLMTIME = ZTIME
ZLUSER = USER
ADDRESS ISPEXEC "LMINIT DATAID(DID) DATASET("SDSN") ENQ(SHRW)"
IF RC > Ø THEN
    DO
        ADDRESS ISPEXEC "SETMSG MSG("ZERRMSG")"
        RETURN 4
    END
ADDRESS ISPEXEC "LMOPEN  DATAID("DID") OPTION(OUTPUT)"
IF RC > Ø THEN
    DO
        ADDRESS ISPEXEC "SETMSG MSG("ZERRMSG")"
        ADDRESS ISPEXEC "LMFREE  DATAID("DID")"
        RETURN 4
    END
DO I = 1 TO CURSTAT.Ø
    LINE = CURSTAT.I
    ADDRESS ISPEXEC "LMPUT DATAID("DID") DATALOC(LINE)",
        "MODE(INVAR) DATALEN("LRECL")"
END
ADDRESS ISPEXEC "CONTROL ERRORS RETURN"
ADDRESS ISPEXEC "LMMREP  DATAID("DID") MEMBER("SNAP") STATS(YES)"
MRC = RC
ADDRESS ISPEXEC "CONTROL ERRORS CANCEL"
ADDRESS ISPEXEC "LMCLOSE DATAID("DID")"
ADDRESS ISPEXEC "LMFREE DATAID(\"DID\")\nIF MRC > 8 THEN
   ADDRESS ISPEXEC "SETMSG MSG(SYNCØ11)"
RETURN Ø
/* OLDLIST. READ THE SNAP MEMBER LIST FOR PDS STATISTICS */
OLDLIST:
  I = Ø
ADDRESS ISPEXEC "LMINIT DATAID(DID) DATASET(\"SDSN\") ENQ(SHR)"
IF RC > Ø THEN
   DO
      ADDRESS ISPEXEC "SETMSG MSG(\"ZERRMSG\")"
      RETURN 4
   END
ADDRESS ISPEXEC "LMOPEN DATAID(\"DID\") OPTION(INPUT)"
IF RC > Ø THEN
   DO
      ADDRESS ISPEXEC "SETMSG MSG(\"ZERRMSG\")"
      ADDRESS ISPEXEC "LMFREE DATAID(\"DID\")"
      RETURN 4
   END
ADDRESS ISPEXEC "LMMFIND DATAID(\"DID\") MEMBER(\"SNAP\")"
IF RC > Ø THEN
   DO
      OLDSTAT.Ø = Ø
      ADDRESS ISPEXEC "SETMSG MSG(SYNCØØ9)"
      ADDRESS ISPEXEC "LMCLOSE DATAID(\"DID\")"
      ADDRESS ISPEXEC "LMFREE DATAID(\"DID\")"
      RETURN Ø
   END
ADDRESS ISPEXEC "LMGET DATAID(\"DID\") MODE(INVAR)\n   \"DATALOC(LINE) DATALEN(LENV) MAXLEN(8Ø)\"
MLC = RC
DO WHILE (MLC = Ø)
   I = I + 1
   OLDSTAT.I = STRIP(LINE)
   ADDRESS ISPEXEC "LMGET DATAID(\"DID\") MODE(INVAR)\n   \"DATALOC(LINE) DATALEN(LENV) MAXLEN(8Ø)\"
      MLC = RC
END
OLDSTAT.Ø = I
ADDRESS ISPEXEC "LMCLOSE DATAID(\"DID\")"
ADDRESS ISPEXEC "LMFREE DATAID(\"DID\")"
RETURN Ø
/* COMPARE, GENERATE NEW/UPDATED MEMBER LIST */
COMPARE:
/* COMPARE CURSTAT TO OLDSTAT, GENERATE NEWSTAT ARRAY */
K = Ø
DO I = 1 TO CURSTAT.Ø
   CALL BINCHOP
   IF FOUND = Ø THEN
      DO

K = K + 1
NEWSTAT.K = CURSTAT.I
END
NEWSTAT.Ø = K
IF NEWSTAT.Ø = Ø THEN
  DO
    ADDRESS ISPEXEC "SETMSG MSG(SYNCØØ5)"
    RETURN 4
  END
END
RETURN Ø
/* BINCHOP, PERFORM BINARY SEARCH IN ARRAY */
BINCHOP:
  TOP = OLDSTAT.Ø
  BOT = 1
  FOUND = Ø
  DO WHILE (FOUND = Ø & TOP >= BOT)
    MID = (TOP + BOT) % 2
    IF CURSTAT.I = OLDSTAT.MID THEN
      FOUND = 1
    ELSE
      IF CURSTAT.I < OLDSTAT.MID THEN
        TOP = MID - 1
      ELSE
        BOT = MID + 1
      END IF
  END
/* IF FOUND THEN MID WAS THE LOCATION IN ARRAY */
RETURN Ø
/* SUBMIT, PRODUCE JOBSTREAM FOR TRANSFER PART 2 */
SUBMIT:
  TABLE = 'SYNC' || ZSCREEN
  ADDRESS ISPEXEC "TBCREATE" TABLE "NAMES(INFO) NOWRITE REPLACE"
  DO I = 1 TO NEWSTAT.Ø
    INFO = NEWSTAT.I
    ADDRESS ISPEXEC "TBADD" TABLE "MULT(5)"
  END
  IF SLST = 'Y' THEN
    DO
      ADDRESS ISPEXEC "TBTOP" TABLE
      ADDRESS ISPEXEC "TBDISPL" TABLE "PANEL(SYNCP3)"
      IF SUBSTR(ZCMD,1,3) = 'CAN' THEN
        DO
          ADDRESS ISPEXEC "SETMSG MSG(SYNCØØ8)"
          ADDRESS ISPEXEC "TBEND" TABLE
          RETURN 4
        END
      END
    ADDRESS ISPEXEC "FTOPEN TEMP"
    ADDRESS ISPEXEC "FTINCL SYNCS1"
    ADDRESS ISPEXEC "FTCLOSE"
ADDRESS ISPEXEC "TBEND" TABLE
ADDRESS ISPEXEC "VGET (ZTEMPF) SHARED"
/* ADDRESS ISPEXEC "BROWSE DATASET('"ZTEMPF"')" */
ADDRESS TSO "SUBMIT '"ZTEMPF"'
RETURN Ø
/* RENAME, RENAME MEMBERS IN PDS (EG BACKUP SNAP MEMBER) */
RENAME:
ARG M1 M2
/* TEST IF M1 EXISTS FIRST */
ADDRESS ISPEXEC "LMINIT DATAID(DID) DATASET("SDSN") ENQ(SHR)"
IF RC > Ø THEN
  DO
    ADDRESS ISPEXEC "SETMSG MSG("ZERRMSG")"
    RETURN 4
  END
ADDRESS ISPEXEC "LMOPEN DATAID("DID") OPTION(INPUT)"
IF RC > Ø THEN
  DO
    ADDRESS ISPEXEC "SETMSG MSG("ZERRMSG")"
ADDRESS ISPEXEC "LMFREE DATAID("DID")"
    RETURN 4
  END
ADDRESS ISPEXEC "LMMFIND DATAID("DID") MEMBER("M1")"
FRC = RC
ADDRESS ISPEXEC "LMCLOSE DATAID("DID")"
ADDRESS ISPEXEC "LMFREE DATAID("DID")"
IF FRC > Ø THEN
  RETURN 4
/* RENAME M1 TO M2 */
ADDRESS ISPEXEC "LMINIT DATAID(DID) DATASET("SDSN") ENQ(SHRW)"
IF RC > Ø THEN
  DO
    ADDRESS ISPEXEC "SETMSG MSG("ZERRMSG")"
    RETURN 4
  END
ADDRESS ISPEXEC "LMOPEN DATAID("DID") OPTION(OUTPUT)"
IF RC > Ø THEN
  DO
    ADDRESS ISPEXEC "SETMSG MSG("ZERRMSG")"
ADDRESS ISPEXEC "LMFREE DATAID("DID")"
    RETURN 4
  END
ADDRESS ISPEXEC "LMMDEL DATAID("DID") MEMBER("M2")"
ADDRESS ISPEXEC "LMMREN DATAID("DID") MEMBER("M1") NEWNAME("M2")"
IF RC > Ø THEN
  DO
ADDRESS ISPEXEC "SETMSG MSG("ZERRMSG")"
ADDRESS ISPEXEC "LMCLOSE DATAID("DID")"
ADDRESS ISPEXEC "LMFREE DATAID("DID")"
    RETURN 4
  END
ADDRESS ISPEXEC "LMCLOSE DATAID(\"DID\")"
ADDRESS ISPEXEC "LMFREE DATAID(\"DID\")"
RETURN Ø
/* SUBROUTINE : EXTRACT TSO SESSION ACCT CODE */
GET_LOGON_ACCT:
X = OUTTRAP('VAR.\')
ADDRESS TSO "LISTUSER" USER "TSO NOR"
X = OUTTRAP('OFF')
SACT = ' '
DO I = 1 TO VAR.Ø WHILE (SACT = ' ')
   IN = INDEX(VAR.I,'ACCTNUM=')
   IF IN > Ø THEN
      SACT = SUBWORD(SUBSTR(VAR.I,IN+8),1,1)
END
RETURN Ø
/* SUBROUTINE : LOCK DATASET FOR SYNC UPDATE TO THIS USER */
SERIAL:
ARG MODE
RES = STRIP(SDSN,'B','\''\'')
IF LENGTH(RES) > 39 THEN
   RES = SUBSTR(RES,1,39)
RES = RES || '.SYNC'
ADDRESS LINK "SYNCL1" MODE RES
RETURN Ø

SYNC2
/* REXX - SYNC2 SYNC FILE TRANSFER PART 2 */
/* THIS RUNS ON THE LOCAL NODE IN BATCH JOB 1 */
/* */
/* PARAMETERS */
/* */
/* STOS=XXXXXX - REMOTE NODE FOR FILE TRANSFER */
/* SDSN=XXXXXXX - FROM DATASET NAME */
/* USER=XXXXXX - USERID */
/* SECL=X - JOB EXECUTION CLASS */
/* SMCL=X - JOB MSG CLASS */
/* SACT=XXXXX - ACCOUNT CODE */
/* ********************************************/ 
ARG PARMS
PARSE VAR PARMS 'STOS=' STOS ' '
PARSE VAR PARMS 'SDSN=' SDSN ' '
PARSE VAR PARMS 'USER=' USER ' '
PARSE VAR PARMS 'SECL=' SECL ' '
PARSE VAR PARMS 'SMCL=' SMCL ' '
PARSE VAR PARMS 'SACT=' SACT ' '
TDSN = SDSN

SDSN = STRIP(SDSN,'B','\''\'')
TDSN = STRIP(TDSN,'B','"')
CALL READ_MEMLIST
JOB = Ø
CNT = Ø
DO J = 1 TO MEMLIST.Ø
   CNT = CNT + 1
   CALL XMIT_MEMBER
END
IF CNT > Ø THEN
   CALL SUBMIT
SAY 'SYNC2 COMPLETED, FILE TRANSFER GENERATED'
EXIT Ø
/* SUBROUTINE : ADD A LINE TO OUTPUT ARRAY */
OUT:
ARG LINE
CUR = CUR + 1
JES.CUR = LINE
RETURN Ø
/* SUBROUTINE : READ MEMBER LIST IN */
READ_MEMLIST:
"EXECIO * DISKR MEMBERS (STEM MEMLIST. FINIS"
IF RC ≠ Ø THEN
   DO
      SAY 'EXECIO READ MEMBER FILE FAILED' RC
      EXIT 4
   END
RETURN Ø
/* SUBROUTINE : GENERATE XMIT FOR MEMBER */
XMIT_MEMBER:
SAY 'MEMBER' MEMLIST.J
IF JOB = Ø THEN
   CALL JOBCARD
   CALL XMIT_DATASET
   CALL XMIT_JCL
   IF CNT > 200 THEN
      CALL SUBMIT
   RETURN Ø
/* SUBROUTINE : GENERATE XMIT DATASET */
MIT_DATASET:
"ALLOC F(TEMP) SP(1 2) CYL NEW LR(ØØ) BLK(232ØØ) RECFM(F B) REUSE"
IF RC ≠ Ø THEN
   DO
      SAY 'ALLOC TEMP XMIT DATA FILE FAILED' RC
      EXIT 4
   END
/* NOTE MAY FAIL IF USER HAS USERID.NAMES.TEXT DATASET ALLOCATED */
X = OUTTRAP('XMS.')
XOPTS = 'NOLOG NOTIFY NOPROLOG NOEPILOG' /* XMIT OPTIONS */
TMEM = STRIP(SUBSTR(MEMLIST.J,1,8))

**XDSN = SDSN || '(' || TMEM || ')'

"XMIT "STOS"."USRI" DATASET('"XDSN"') OUTDD(TEMP)" XOPTS

IF RC ≠ 0 THEN

DO

   SAY 'XMIT TO CREATE DATA FILE FAILED' RC

   DO K = 1 TO XMS.Ø

      SAY XMS.K

   END

END

EXIT 4

END

X = OUTTRAP('OFF')

"EXECIO * DISKR TEMP (STEM DAT. FINIS"

IF RC ≠ 0 THEN

DO

   SAY 'EXECIO READ XMIT DATA FAILED' RC

   EXIT 4

END

"FREE F(TEMP)"

/* CHECK THAT THE DELIMITER DOES NOT OCCUR IN THE XMIT DATA */

DLA.Ø = 7

DLA.1 = '#'

DLA.2 = '@'

DLA.3 = '@'

DLA.4 = '@#'

DLA.5 = '@@'

DLA.6 = '##'

DLA.7 = '

DOK = Ø

DO K = 1 TO DLA.Ø WHILE (DOK = Ø)

   DLM = DLA.K

   DOK = 1

   DO L = 1 TO DAT.Ø WHILE (DOK = 1)

      IF SUBSTR(DAT.L,1,2) = DLM THEN

         DOK = Ø

      END

END

IF DOK = Ø THEN

DO

   SAY 'XMIT DATA CONTAINED ALL POSSIBLE DELIMITERS, JOB NOT SENT'

   EXIT 4

END

RETURN Ø

/* SUBROUTINE: GENERATE JOBCARD */

JOBCARD:

CUR = Ø

CALL OUT "//"USER"XR JOB ("SACT"),"USER",CLASS="SECL","USER":""

CALL OUT "// NOTIF"USER":"

CALL OUT "// MSGLEVEL=(1,Ø),MSGCLASS="SMCL"

CALL OUT "//**"

CALL OUT "*/XEQ "STOS

© 1998. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (940) 455 7050, fax (940) 455 2492.
CALL OUT "/**"
CALL OUT "/** SYNC2 INITIATED FILE TRANSFER"
CALL OUT "/** FIRST, VERIFY DATASET EXISTS"
CALL OUT "/**"
CALL OUT "/*VERIFY EXEC PGM=IEFBR14"
CALL OUT "/*DATASET DD DISP=SHR,DSN="TDSN"
JOB = 1
RETURN Ø
/* SUBROUTINE : GENERATE XMIT JCL */
XMIT_JCL:
/* NOTE JOB ACHIEVES SERIALISATION BY ENQ WITHIN SYNC3 ON DSN */
CALL OUT "/** NOW, RECEIVE" TDSN TMEM
CALL OUT "/**"
CALL OUT "/*RECEIVE EXEC PGM=IKJEFT01,DYNAMNBR=30,COND=(4,LT)"
CALL OUT "/*SYSPROC DD DSN=SYS1.REXX.LIB,DISP=SHR"
CALL OUT "/*SYSTSIN DD SYSOUT=""
CALL OUT "/*SYNC3" TDSN TMEM
CALL OUT "/*RECEIN DD DATA,DLM="DLM"
DO K = 1 TO DAT.Ø /* IMBED THE XMIT DATA */
  CUR = CUR + 1
  JES.CUR = DAT.K
END
CALL OUT DLM /* TERMINATE THE DATA */
RETURN Ø
/* SUBROUTINE : SUBMIT JOB, SEVERAL MAY BE GENERATED */
SUBMIT:
JES.Ø = CUR
"ALLOC F(JES2) SYSO(P) WRIT(INTRDR) LR(Ø0) BLK(23200)"
  "RECFM(F B) REUSE"
IF RC ≠ Ø THEN
  DO
    SAY 'ALLOC JES2 DATA FILE FAILED' RC
    EXIT 4
  END
"EXECIO * DISKW JES2 (STEM JES. FINIS" /* WRITE JCL TO INTRDR */
IF RC ≠ Ø THEN
  DO
    SAY 'EXECIO WRITE TO DD JES2 FAILED' RC
    EXIT 4
  END
"FREE F(JES2)"
DROP JES.
CNT = 0
JOB = Ø
RETURN Ø
/* REXX - SYNC3 TSO AUTO FILE TRANSFER RECEIVE */
/* THIS RUNS ON THE REMOTE NODE TO RECEIVE DATA */
/* */
/* RECEIVE WITH ENQ SERIALIZATION ON DATASET USING */
/* SYNCL1 TO ISSUE SPFEDIT ENQ AND DEQ EXCLUSIVE. */
/* THE ENQ PROTECTS THE DATASET AGAINST MULTI-WRITE */
/* WITHOUT REQUIRING DISP=OLD ALLOCATION (LIKE ISPF). */
/* *******************************************/
ARG DSN MEM
DSN = STRIP(DSN)
MEM = STRIP(MEM)
STATUS = SYSDSN("''DSN''")
IF STATUS ^= 'OK' THEN
   DISP = 'NEW' /* ADD UNIT OPERAND IF REQUIRED TO ALLOCATE */
ELSE
   DISP = 'SHR'
OPTS = 'SYSOUT(Ø)' /* NULL SYSOUT CLASS FOR IEBCOPY MESSAGES */
"DELSTACK"
X = PROMPT('ON')
ADDRESS LINK "SYNCL1 ENQ" DSN
IF RC > Ø THEN
   DO
      SAY 'ENQ FAILED' RC
      EXIT RC
   END
   IF MEM = '' THEN
      QUEUE "DA("''DSN''")" DISP OPTS
   ELSE
      QUEUE "DA("''DSN"("MEM")")" DISP OPTS
   QUEUE "END"
   ADDRESS TSO "RECEIVE INDDNAME(RECIN) LOGDS('NULLFILE') NONAMES"
XRC = RC
IF RC > Ø & RC <= 4 THEN
   DO
      SAY 'RECEIVE WARNING' RC
   END
   IF RC > 4 | RC < Ø THEN
      DO
         SAY 'RECEIVE FAILED' RC
      END
ADDRESS LINK "SYNCL1 DEQ" DSN
IF RC > Ø THEN
   DO
      SAY 'DEQ FAILED' RC
   END
"DELSTACK"
EXIT XRC
ISPF PANEL SYNCP1 SOURCE

)ATTR
    # TYPE(INPUT) INTENS(NON) CAPS(ON) JUST(LEFT)
    $ TYPE(TEXT) INTENS(LOW) COLOR(YELLOW)
)BODY

%-------------------------------%PDS SYNCHRONISATION%-------------------------------
+-
%OPTION ---->_ZCMD
+-
%1$ TRANSMIT+- Send changed members since last snapshot and take new snapshot
%2$ NEW SNAP+- Record a snapshot of member statistics (replace existing snap)
%3$ OLD SNAP+- Rename previous snapshot to recorded snapshot (if xmit failed)
%4$ DIS SNAP+- Display recorded snapshot member statistics (if any)
%5$ DEL SNAP+- Delete the recorded snapshot member statistics (if any)
+-
+NOTE : Only members with ISPF (or PDSMAN) statistics will be processed.
+Specify 'DATASET' in TSO syntax (do not specify a member name)
%DATASET NAME %---->_SDSN+(On both systems)
+Review xmit  %---->_SLST+(Y/N, to review changed list before transmit)
+-
%To+MVS node  %---->_STOS + (second job will execute here)
+-
+Batch JOBs+(jobname &ZUSER.XF/R) First job generates xmit, second receives it
+JOB Acct  %---->_SACT +
+JOB Class  %---->_SECL+ JOB msgclass %---->_SMCL+
+-
+Enter details and press%ENTER+to continue or press%END+to exit
)INIT
.HELP = SYNCP2
&ZCMD = ''

© Xephon 1998

WLM in an sysplex environment

We are working on a parallel sysplex project. After several months of tests and studies on a dedicated test platform, we are implementing sysplex on our operational MVS systems. At the beginning of our studies, in our site, we had five operational MVS systems:

- **PMVS** – for production activity
- **JMVS and TMVS** – for development activities
• RMVS – for test and integration activity (pre-production)
• CMVS – for our datawarehouse activity.

All these systems are considered (even if they are not production systems) as operational systems: we use OPC/ESA to schedule administration jobs, we share tapes, libraries, and disks among all these systems. With sysplex, it is possible, if you integrate all your systems in the same sysplex, to use new sysplex-wide facilities: XCF support for OPC/ESA, automatic tape switching (IEFAUTO), shared logic, HCD sysplex-wide activation.

And we think that in the near future more and more system facilities will use sysplex architecture. Sysplex will help use to manage our MVS images. This is why we have decided to integrate all our existing (and future) MVS systems in the same sysplex.

To implement parallel sysplex we have decided, for test and validation reasons, to clone first our pre-production system RMVS with a new MVS image ZMVS. In the same way, our production system PMVS will be cloned with a new image FMVS. So, at the end of the project, our sysplex will include seven MVS systems.

LOGICAL MVS IMAGE NOTION

We will group systems that will participate in the same activity (real parallel sysplex clones) in a new notion, a MVS logical image, ie:

• PMVS and FMVS will be grouped in a logical MVS, MMVS.
• RMVS and ZMVS in another logical MVS, NMVS.

We will implement MVS cloning, DB2 data sharing, and CICS cloning on these logical MVSES.

WLM, SYSPLEX, AND SMFID

In a sysplex working in WLM goal mode, the same WLM policy will be valid for all systems participating in the sysplex. So, our seven MVS images will have to use the same WLM policy. But, a TSO transaction in one of our development systems doesn’t have the same profile as a TSO transaction in our production systems. Worse WLM
will manage service classes at sysplex level. So, it will compute sysplex global response time. For example, let us say that a response time of 1 second is desired in a sysplex of two systems, A and B.

If on system A (development system) you have 9000 TSO transactions per hour with an average response time of 0.9 seconds and on system B 1000 TSO transactions with a response time of 1.5 seconds. At the sysplex level, the response time seen by WLM is:

\[
(9000 \times 0.9 + 1000 \times 1.5) / 10000 = (81000 + 15000)/10000 = 0.96 \text{ s}
\]

And WLM is happy. But B users aren’t at all. What is true for TSO is also true for batch jobs and STCs.

It is not possible to implement WLM in that type of sysplex (where you find different kinds of activities) because WLM doesn’t allow you to use SMFID (or SYSNAME) in classification rules.

WLMRESET FACILITY

To be able to assign a specific service class for specific systems we had to write a little facility to automatically reset to specific service classes jobs, STCs or TSO users when they start. In WLM, for each service class, we define:

- A ‘shared’ service class starting with a $ (eg: $_TSO_30) referenced by classification rules.
- A specific service class for each ‘logical MVS’ starting with the first letter of the SMFID (eg: T_TSO_30 for TMVS).

For example, for standard TSO users, we defined six service classes:

- $_TSO_30 used in classification rules.
- N_TSO_30 for NMVS logical MVS (PMVS + FMVS).
- M_TSO_30 for MMVS logical MVS (RMVS + ZMVS).
- T_TSO_30 for TMVS image.
- C_TSO_30 for CMVS image.
- J_TSO_30 for JMVS image.
WLMRESET, which is started as an STC on each MVS image, works as an Extended MCS console, which traps messages IEF403I and IEF125I and resets automatically the starting task to the desired service class replacing the first letter. The program was written on an OS/390 1.3 system. WLMRESET uses several interesting functions:

- Console communication.
- Extended MCS console.
- ENF Listener – WLMRESET is listening for WLM policy activations.
- SYSEVENT interface.

WLMRESET

WLMRESET CSECT ,
WLMRESET AMODE 31
WLMRESET RMODE ANY
*
* BEGIN AR MODE LINKAGE CONVENTION
  BAKR   R14,Ø       SAVE REGS
  SAC    512          SET AR MODE
  SYSSSTATE ASCENV=AR LET MACROS KNOW
  LAE    R12,Ø(R15,Ø) BASE AND ADDRESS REGS
  USING  WLMRESET,R12 ADDRESSABILITY
  STORAGE OBTAIN,LENGTH=DYNL GET DYNAMIC STORAGE
  LAE    R11,Ø(Ø,R1) USE R11 AS DYN BASE
  LAE    R2,DYNMODEL ADDRESS OF DYNAMIC AREA MODEL
  L      R3,=A(DYNL) LENGTH OF DYNAMIC AREA
  LAE    R4,Ø(Ø,R11) ADDRESS OF DYNAMIC AREA
  LR     R5,R3 LENGTH OF DYNAMIC AREA
  MVCL   R4,R2 COPY MODEL TO DYNAMIC AREA
  USING  DYNMODEL,R11 MAP MODEL OVER DYNAMIC AREA
  LAE    R13,SV PUT SAVE AREA ADDR IN R13
  MVC    4(R13),=C'F1SA' SET ACRO IN SAVE AREA
  ST     R12,BASE_REG
* END AR MODE LINKAGE CONVENTION
* BEGIN INITIALIZATION
  SAC    0 SET PRIMARY MODE
  SYSSSTATE ASCENV=P LET MACROS KNOW
  LA     R9,COMADDR GET ADDRESS FOR COM AREA
  EXTRACT (R9),FIELDS=COMM, EXTRACT THE COM AREA
  MF=(E,EXTRACT)
  L      R9,COMADDR GET ADDRESS OF THE AREA
  USING  COM,R9 USE R9 AS BASE ADDRESS OF COMM AREA
ICM     R7,15,COMCIBPT     GET ADDRESS OF THE CIB  
BZ      NOCIB              NO START CIB  
BAL     R14,DOCIB          PROCESS THE CIB  
NOCIB  DS      ØH  
QEDIT  ORIGIN=COMCIBPT,   X  
        CIBCTR=1 SET MODIFY LIMIT TO 1  
L      R1,COMECEBPB GET ADDRESS OF THE COM ECB  
O      R1,'=8ØØØØØØ' SET HIGH BIT - LAST ECB IN LIST  
ST     R1,MODECBP PUT ADDR OF MODIFY ECB IN LIST  
LA     R1,ECBP GET ADDR OF MESSAGE ECB  
ST     R1,MEBP PUT INTO ECB LIST  
LA     R1,ALERT GET ADDR OF ALERT ECB  
ST     R1,AECBP PUT INTO ECB LIST  
MVC    WTOID,=C'WJS999I ' MESSAGE ID FOR ECHOED MESSAGES  
MVC    COMETSP,STRTR STARTED TASK, INIT MSG BACK TO CONS  
LA     R1,INITMSG GET INITIALIZATION MESSAGE  
BAL     R14,MESSR DISPLAY MESSAGE  
MVI     CMDRSP,Ø MAKE SURE COMMAND RESPONSE RESET  
WTO    TEXT=INITMS2, DISPLAY HELD INITIALIZATION MSG  
        MF=(E,WTOHOLD) X  
ST     R1,MSGID KEEP HELD MESSAGE ID FOR DOM  
* END INITIALIZATION  
BAL     R14,GETSYS  
BAL     R14,INIENTF  
BAL     R14,ACTCON  
BAL     R14,REFRES  
*="LOOP:  ASCMODE=PRIMARY, IN-LINE ENTRY  
* FUNCTION: MAIN PROCESSING LOOP; WATCH FOR DONE AND POSTED ECBS  
* OPERATION:  
* IF DONE THEN EXIT  
* WAIT FOR ECB POST (MESSAGE, ALERT, OR MODIFY/STOP)  
* IF MESSAGE ECB POSTED, CALL GETMSGS  
* IF ALERT ECB POSTED, CALL DOALERT  
* IF MODIFY/STOP ECB POSTED, CALL DOCIB  
* GO BACK TO TOP OF LOOP  
*="LOOP:  ASCMODE=PRIMARY, IN-LINE ENTRY  
* FUNCTION: MAIN PROCESSING LOOP; WATCH FOR DONE AND POSTED ECBS  
* OPERATION:  
* IF DONE THEN EXIT  
* WAIT FOR ECB POST (MESSAGE, ALERT, OR MODIFY/STOP)  
* IF MESSAGE ECB POSTED, CALL GETMSGS  
* IF ALERT ECB POSTED, CALL DOALERT  
* IF MODIFY/STOP ECB POSTED, CALL DOCIB  
* GO BACK TO TOP OF LOOP  
SAC     Ø SET PRIMARY MODE FOR MODESET  
SYSSTATE ASCENV=P LET MACROS KNOW PRIMARY MODE  
LOOPS  DS      ØH MAIN PROCESSING LOOP  
CLI     DONE,Ø CHECK FOR TERMINATION  
BZ      WAIT NO, DO WAIT  
MODESET MF=(E,SUPØ) SET SUP STATE AND KEY ZERO  
L      R1,ENFEXITA  
FREEMAIN R,LV=ENFEXITL,A=(R1),SP=241  
MODESET MF=(E,PROB)  
* DETACH TCBADDR  
STORAGE RELEASE, FREE DYNAMIC STORAGE  
        LENGTH=DYNL,  
        ADDR=(R11)
PR  EXIT PROGRAM

WAIT  DS  ØH
WAIT ECBLIST=ECBS  WAIT FOR A MESSAGE/ALERT/MODIFY/STOP
L  R1,ECB  GET MSG ECB
N  R1,=X'4Ø000000'  CHECK FOR POST
BZ  CKALRT  NOT SET, CHECK ALERT
XC  ECB,ECB  CLEAR MESSAGE ECB
BAL  R14,GETMSGS  PROCESS THE MESSAGE

CKALRT  DS  ØH
L  R1,ALERT  GET ALERT ECB
N  R1,=X'4Ø000000'  CHECK FOR POST
BZ  CKCIB  NOT POSTED, CHECK MODIFY
XC  ALERT,ALERT  CLEAR ALERT ECB
BAL  R14,DOALERT  PROCESS ALERT

CKCIB  DS  ØH
ICM  R1,15,COMCIBPT  GET CIB POINTER
BZ  LOOP  NO CIB, BACK TO MAIN LOOP
BAL  R14,DOCIB  PROCESS THE CIB (QEDIT TAKES CARE OF X
THE ECB)
B  LOOP  BACK TO MAIN LOOP

* ——————————————————————————————————— *
* - GETMSGS: BRANCH ENTERED ASCMODE=PRIMARY, SETS ASCMODE=AR    - *
* - FUNCTION: PROCESS ALL MESSAGES QUEUED TO THIS CONSOLE        - *
* - OPERATION:                                                      - *
* - INVOKE MCSOPMSG IN SUPERVISOR STATE                            - *
* - WHEN A MESSAGE IS RETURNED (GOTMDB)                            - *
* - LOOP THROUGH THE MDB OBJECTS                                   - *
* - WHEN GENERAL OBJECT, CALL GOTMDBG                              - *
* - WHEN CONTROL PROG OBJECT, CALL GOTMDBDC                       - *
* - WHEN TEXT OBJECT, CALL GOTMDBT                                 - *
* - OTHERWISE UNKNOWN OBJECT TYPE                                  - *
* - WHEN AN ERROR OCCURS IN MCSOPMSG (GOTERR)                      - *
* - PUT OUT ERROR MESSAGE                                           - *
* - SET DONE FLAG TO EXIT PROGRAM                                  - *
* - RETURN TO CALLER                                               - *
* ——————————————————————————————————— *

SYSSTATE ASCENV=P  LET MACROS KNOW PRIMARY MODE
GETMSGS  DS  ØH
BAKR  R14,Ø  SAVE CALLER ENVIRONMENT

MSGLP  DS  ØH
SAC  0  SET PRIMARY MODE FOR MODESET
MODESET MF=(E,SUP)  SET SUP STATE
SAC  512  THIS PROCEDURE RUNS IN AR MODE
SYSSTATE ASCENV=AR  LET MACROS KNOW
MCSOPMSG REQUEST=GETMSGS, GET A MESSAGE  X
CONSID=CNID, MY CONSOLE ID  X
RTNCODE=RC, SAVE RETURN CODE  X
RSNCODE=RSN, SAVE REASON CODE  X
MF=(E,MCSOPMPL)
LAE  R8,Ø(Ø,R1)  PUT MDB ADDRESS IN R8
USING MDB,R8 ADDRESSABILITY TO THE MDB
SAC Ø SET PRIMARY MODE FOR MODESET
SYSSTATE ASCENV=P LET MACRO KNOW
MODESET MF=(E,PROB) SET PROBLEM STATE
SAC 512 THIS PROCEDURE RUNS IN AR MODE
SYSSTATE ASCENV=AR LET MACROS KNOW
MVI MDBFLGS,Ø CLEAR PROCESSING FLAGS
MVI CDRSP,Ø ASSUME NOT ISSUING COMMAND RESPONSE
LA R15,8 LOOKING FOR MESSAGE RETURNED
C R15,RC SEE IF ANY MESSAGES
BH GOTMDB PROCESS IT (RC<8)
BL GOTERR SOME KIND OF ERROR (RC>8)
PR NO MORE MESSAGES (RC=8)
GOTERR DS ØH
LA R1,BADGET GET ERROR MESSAGE
BAL R14,MESSR SHOW IT
MVI DONE,1 SET DONE FLAG
PR RETURN ERROR MESSAGE

* ——————————————————————————————————— *
* - GOTMDB: ENTRY VIA BRANCH (NOT A SUBROUTINE)                     -*
* - FUNCTION: PROCESS THE GENERAL OBJECT AND CONTROL PROGRAM OBJECT   -*
* - FOR A MESSAGE. ASSUMPTIONS MUST NOT BE MADE THAT THESE -*
* - OBJECTS WILL PRECE ANY TEXT OBJECTS.                          -*
* ——————————————————————————————————— *
GOTMDB DS ØH
LR R5,R8 CALC END OF MBD IN R5
AH R5,MDBLEN START+MDBLEN IN HEADER
LR R6,R8 REMEMBER START OF MDB FOR PASS 2
LA R8,MDBHLEN(R8) BUMP TO 1ST OBJECT
OBJLP DS ØH LOOP THROUGH THE OBJECTS
LH R3,MDBTYPE GET TYPE
C R3,=A(MDBGOBJ) CHECK FOR GENERAL OBJECT
BNE NOTG NOT GENERAL OBJECT
TM MDBFLGS,MEBFGO SEE IF FIRST GENERAL OBJECT
BO NXTOBJ NO, SKIP IT
BAL R14,GOTMDB PROCESS GENERAL OBJECT
B NXTOBJ BUMP TO NEXT OBJECT

NOTG DS ØH
C R3,=A(MDBCOBJ) CHECK FOR CONTROL PROG OBJECT
BNE NOTC NOT CONTROL PROG OBJECT
TM MDBFLGS,MDBFCO SEE IF FIRST CONTROL PROG OBJECT
BO NXTOBJ NO, SKIP IT
BAL R14,GOTMDBC PROCESS CONTROL PROG OBJECT
B NXTOBJ BUMP TO NEXT OBJECT
NOTC DS ØH NOT CONTROL PROG OBJ
NXTOBJ DS ØH FIND NEXT OBJECT
TM MDBFLGS,MDBFCO+MDBFCO SEE IF WE FOUND GENERAL AND SCP
BO FNDTXT GOT THEM, LOOP THROUGH TEXT OBJS
AH R8,MDBLEN BUMP TO NEXT OBJECT
CR R8,R5 SEE IF THIS IS THE END
BL OBJLP NO, GET ANOTHER OBJECT
B MSGLP MISSING NECESSARY OBJECTS, SKIP IT

* ———————————————————————————————————*
* - FNDTXT: ENTRY VIA BRANCH (NOT A SUBROUTINE) -*
* - FUNCTION: PROCESS ALL TEXT OBJECTS IN ALL MDBS FOR THIS MESSAGE. -*
* - TEXT OBJECTS ARE ALWAYS ORDERED, BUT IT CANNOT BE -*
* - ASSUMED THAT THEY ARE CONTIGUOUS. -*
* - OPERATION: -*
* - FIND END OF MDB -*
* - GET POINTER TO NEXT MDB IN MESSAGE -*
* - LOOP THROUGH MDBS -*
* - LOOP THROUGH OBJECTS -*
* - WHEN TEXT OBJECT -*
* - CALL GOTMDBT TO PROCESS TEXT OBJECT -*
* - OTHERWISE IGNORE OBJECT -*
* - SKIP TO NEXT OBJECT -*
* - ADD OBJECT LENGTH -*
* - IF END OF MDB, MOVE TO NEXT MDB -*
* ———————————————————————————————————*
FNDTXT DS ØH
LR R8,R6 RESET R8 TO START OF MDB
TXTLP DS ØH
LR R5,R8 CALC END OF MDB IN R5
AH R5,MDBLEN START+MDBLEN IN HEADER
LAE R6,Ø(Ø,R8) CALC PREFIX ADDRESS IN R6
SH R6,=AL2(MDBPLNNO) PREFIX=START-PREFIX LENGTH
USING MDBPRFX,R6 GET ADDRESSABILITY
L R6,MDBPNEXT GET FORWARD POINTER IN R6
DROP R6 R6 NO LONGER BASE FOR PREFIX
LA R8,MDBHLEN(R8) BUMP TO 1ST OBJECT
TOBJLP DS ØH LOOP THROUGH THE OBJECTS
LH R3,MDBTYPE GET TYPE
C R3,=A(MDBTOBJ) CHECK FOR TEXT OBJECT
BNE NOTT NOT TEXT OBJECT
BAL R14,GOTMDBT PROCESS TEXT OBJECT
NOTT DS ØH
AH R8,MDBLEN BUMP TO NEXT OBJECT
```
CR     R8,R5        SEE IF THIS IS THE END
BL     TOBJLP       NO, GET ANOTHER OBJECT
LTR    R6,R6        CHECK FOR MORE MDBS FOR MESSAGE
BZ     MSGLP         DONE WITH MESSAGE
LR     R8,R6        NEXT MDB
B      TXTLP         PROCESS THE MDB
DROP   R8

*——————————————————————————————————*
*- GOTMDBG: BRANCH ENTERED, ASCMODE=AR, R8=ADDR(GENERAL OBJECT) -*
*- FUNCTION: PROCESS MDB GENERAL OBJECT -*
*- OPERATION: -*
*- ESTABLISH ADDRESSABILITY TO THE GENERAL OBJECT -*
*- INDICATE GENERAL OBJECT PROCESSED -*
*——————————————————————————————————*

SYSSTATE ASCENV=AR      LET MACROS KNOW AR MODE
GOTMDBG DS ØH
BAKR   R14,Ø       SAVE CALLER ENVIRONMENT
USING  MDBG,R8     ADDRESSABILITY TO GENERAL OBJECT
OI     MDBFLGS,MDBFGO SET PROCESSED GENERAL OBJECT
PR
DROP   R8

*——————————————————————————————————*
*- GOTMDBC: BRANCH ENTERED, ASCMODE=AR, R8=ADDR(CONTROL PROG OBJECT)-*
*- FUNCTION: PROCESS MDB CONTROL PROGRAM OBJECT -*
*- OPERATION: -*
*- ESTABLISH ADDRESSABILITY TO THE CONTROL PROGRAM OBJECT -*
*- IF THIS IS AN MVS OBJECT -*
*- SET FLAG INDICATING CONTROL PROG OBJECT FOUND FOR THE MSG -*
*- SAVE MESSAGE TEXT OFFSET FOR TEXT PROCESSING -*
*- IF THIS IS A COMMAND RESPONSE MESSAGE -*
*- SAVE THE CART -*
*- INDICATE THAT THE TEXT ECHO SHOULD BE COMMAND RESPONSE -*
*——————————————————————————————————*

SYSSTATE ASCENV=AR      LET MACROS KNOW AR MODE
GOTMDBC DS ØH
BAKR   R14,Ø       SAVE CALLER ENVIRONMENT
USING  MDBSCP,R8   ADDRESSABILITY TO CONTROL PROG OBJECT
CLC    MDBCPNAM,=C'MVS '  MAKE SURE IT IS AN MVS OBJECT
BNE    GOTC1        IF NOT, JUST SKIP IT
MVC    XJOB,MDBC0JBN
MVC    XASID,MDBCASID
OI     MDBFLGS,MDBFCO SET PROCESSED CONTROL PROG OBJECT
LH     R1,MBCTOFF    GET TEXT OFFSET
ST     R1,TOFF      SAVE IT FOR TEXT PROCESSING
TM     MDBCATT1,MDBCMCS C  CHECK IF COMMAND RESPONSE
BZ     GOTC1        NOT COMMAND RESPONSE
MVC    MCART,MDBCCART HOLD ONTO CART
MVI    CMDRSP,1     ISSUE ANY WTOS AS CMD RESPONSE
GOTC1  DS ØH
PR
```
DROP R8

* - GOTMDBT: BRANCH ENTERED, ASCMODE=AR, R8=ADDR(TEXT OBJECT) *
* - FUNCTION: PROCESS MDB TEXT OBJECTS *
* - OPERATION: *
* - ESTABLISH ADDRESSABILITY TO THE TEXT OBJECT *
* - CALCULATE THE LENGTH OF THE TEXT *
* - MOVE IT TO A BUFFER *
* - SET THE LENGTH *
* - ISSUE TEXT AS A SINGLE LINE WTO *

SYSSTATE ASCENV=AR         LET MACROS KNOW AR MODE
GOTMDBT DS ØH
BAKR R14,Ø              SAVE CALLER ENVIRONMENT
USING MDBT,R8            ADDRESSABILITY TO TEXT OBJECT
LH R1,MDBTLEN           GET TEXT OBJECT LENGTH
S R1,=A(MDBTMSGT-MDBTLEN) SUBTRACT NON-TEXT SIZE
S R1,TOFF              TAKE OFF OFFSET TO TEXT
C R1,=A(L'WTOTXT)     MAKE SURE ITS NOT TOO LONG FOR BUF
BNH GOTT1            OK
L R1,=A(L'WTOTXT)  NOT OK, TRUNCATE AT BUF LENGTH
GOTT1 DS ØH
S R1,=F'1'            SET UP FOR MVC
LAE R2,MDBTMSGT        GET ADDRESS OF TEXT
A R2,TOFF             BUMP PAST PREFIX INFO
CLC WTOID,Ø(R2)        SEE IF THIS MESSAGE IS MY ECHO
BE GOTTX            DON'T REDISPLAY MY TEXT ECHO
CLC IEF403I,Ø(R2)      SEE IF THIS MESSAGE IS IEF403I
BNE NXTMSG
GOTT2 DS ØH
EX R1,GOTTMVC        MOVE TEXT TO BUFFER
A R1,=A(L'WTOTXT+1) CALC LENGTH FOR WTO
STH R1,WTOBUF       SET MESSAGE LENGTH
LA R1,WTOBUF       GET BUF ADDR
BAL R14,MESSTR    DISPLAY THE TEXT
GOTTX DS ØH
*
RESET DS ØH
BAKR R14,Ø              SAVE CALLER ENVIRONMENT
ESTAEX RECOVERY,CT,PARAM=(R11) USING WORK AREA AS PARAMETER
SAC Ø                 RUN IN PRIMARY MODE
SYSSTATE ASCENV=P       TELL MACROS PRIMARY MODE
MODESET MF=(E,SUPØ) SET SUP STATE AND KEY ZERO
*====================================================================
* GET SRVCLASS FROM INCOMING WORK
*=====================================================================

CLC XJOB,=CL8'INIT'
BE RESETOK
CLC XJOB,=CL8'ASCHINT'
BE RESETOK
SR R4,R4
LH R4,XASID LOAD ASID FOR SYSEVENT
SR R5,R5
LA R5,Ø2Ø MAXIMUM NUMBER OF LOOPS USING OUCB,R6

WAITEX EQU *
TM OUCBTFL,OUCBINC INITIATOR ATTACH CURRENT ?
BO TRANEX YES, GO TO RESET
TM OUCBTFL,OUCBNTR TRANSACTION ENDING ?
BO RESETOK YES, DON'T RESET
STIMER WAIT,DINTVL=INT NO, WAIT A LITTLE BIT
BCT R5,WAITEX AND TRY AGAIN
MVC WTO(WTOL),WTOC
MVC WTO+Ø4(Ø8),=CL8Ø'WJSØ14I UNABLE TO GET SRVCLASS FOR '
MVC WTO+4Ø(Ø8),XJOB
WTO MF=(E,WTO)

B RESETOK

TRANEX EQU *
LA R3,MYRASD
USING RASD,R3
LA R1,MYRASD
LA R2,RASD_LEN
STH R2,RASDL LEN
SYSEVENT REQASD,ASID=(R4),ENTRY=BRANCH
CLC RASDSCL,=CL8'SYSTEM' DON'T RESET SYSTEM SRVCLASS
BE RESETOK
CLC RASDSCL,=CL8'SYSSTC' DON'T RESET SYSSTC SRVCLASS
BE RESETOK

MVSERV EQU *
MVC XSRV,RASDSCL MOVE SRVCLASS
MVC XSRV(1),XSYSL OVERIDE $

*====================================================================
* RESET SRVCLASS
*=====================================================================

RESETL EQU *
LA R1,XSRV
SYSEVENT RESETPG,ASID=(R4),ENTRY=BRANCH,TYPE=SRVCLASS

RETRYPT EQU *
RESETOK EQU *
ESTAEX Ø
MDESET MF=(E,PROB) SET PROB STATE
PR

*============================================*
*              LOOP TO RESET ALL ADDRESS SPACES               *
*============================================*

REFRES DS 0H
BAKR R14,Ø   SAVE CALLER ENVIRONMENT
SAC Ø       RUN IN PRIMARY MODE
SYSTYPE ASCENV=P TELL MACROS PRIMARY MODE

*============================================*
*              LOOK FOR ASID FOR SYSEVENT MACRO            *
*============================================*

SR R4,R4
USING PSA,R4
L R5,FLCCVT
USING CVTMAP,R5
L R6,CVTASVT
USING ASVT,R6
DROP R4
L R4,ASVTMAXU LOAD MAX NUMBER OF ASCB ENTRIES
DROP R5
LA R5,ASVTENTY ASCB ADDRESS SLOT
RASCB EQU *
L R7,Ø(R5) ADDRESS OF ASCB
USING ASCB,R7
TM R3,ASVTAVAL AVAILABLE ENTRY ?
BO REXT GO TO NEXT ENTRY
CLC ASCBASCB,=CL4'ASCB' VALID ASCB ?
BNE REXT
L R6,ASCBOUCB
USING OUCB,R6
L R8,ASCBJBNI CASE OF A JOB
LTR R8,R8
BZ RROC
RROC EQU *
L R8,ASCBJBNS CASE OF A STC OR A TSO
LTR R8,R8
BZ RESTJ
RESTJ EQU *
MVC XJOB,Ø(R8)
MVC XASID,ASCBASID GET ASID
BAL R14,RESET
REXT EQU *
LA R5,4(R5) POINT TO NEXT ENTRY IN ASVT
BCT R4,RASCB
MODESET MF=(E,PROB) SET PROB STATE
PR

SASID DS 0H
BAKR R14,Ø  SAVE CALLER ENVIRONMENT
SAC Ø       RUN IN PRIMARY MODE
SYSTYPE ASCENV=P TELL MACROS PRIMARY MODE
SR R4,R4
USING PSA,R4
L R5,FLCCVT
USING CVTMAP,R5
L R6,CVTASVT
USING ASVT,R6
DROP R4
L R4,ASVTMAXU
DROP R5
LA R5,ASVTENTY
LOAD MAX NUMBER OF ASCB ENTRIES
MSCB EQU *
L R7,Ø(R5)
ADDRESS OF ASCB
USING ASCB,R7
TM R3,ASVTAVAL
AVAILABLE ENTRY ?
BO NEXT
GO TO NEXT ENTRY
CLC ASCBASCB,=CL4'ASCB'
VALID ASCB ?
BNE NEXT
L R8,ASCBJBNI
CASE OF A JOB
LTR R8,R8
BZ PROC
B TESTJ
PROC EQU *
L R8,ASCBJBNS
CASE OF A STC OR A TSO
LTR R8,R8
BZ TESTJ
TESTJ EQU *
CLC XJOB,Ø(R8)
BNE NEXT
L R6,ASCBOUCB
USING OUCB,R6
MVC XJOB,Ø(R8)
MVC XASID,ASCBASID
GET ASID
BAL R14,RESET
NEXT EQU *
LA R5,4(R5)
POINT TO NEXT ENTRY IN ASVT
BCT R4,MASCB
MODESET MF=(E,PROB)
SET PROB STATE
PR
GOTTMVC DS ØH
MVC WTOTXT(Ø),Ø(R2)
DROP R8
*------------------------------------------*
* LOOK FOR ASID FOR SYSEVENT MACRO    =
*------------------------------------------*

*— DOALERT: BRANCH ENTERED ASCMODE=PRIMARY, SETS ASCMODE=AR    -*
*— FUNCTION: PROCESS A CONSOLE ALERT NOTIFICATION    -*
*— OPERATION:    -*
*— ESTABLISH ADDRESSABILITY THE THE CONSOLE STATUS AREA    -*
*— CHECK EACH ALERT INDICATOR    -*

* IF SET, PUT OUT A MESSAGE *
* NO ERROR HANDLING IS PERFORMED THIS EXAMPLE, JUST *
* DEACTIVATE THE CONSOLE ON ANY ALERT *

DOALERT DS ØH
  BAKR R14,Ø  SAVE CALLER ENVIRONMENT
  SAC 512   GET INTO AR MODE
  SYSSTATE ASCENV=AR LET MACROS KNOW
  L  R2,CSA GET ADDRESS OF THE STATUS AREA
  LAM  R2,R2,CSAALET GET ALET FOR STATUS AREA
  USING MCSCSA,R2 ESTABLISH ADDRESSABILITY
  CLI MCSCMLIM,Ø REACHED MEMORY LIMIT?
  BZ ALRT1 NO
  LA R1,MSGMLIM GET ERROR MESSAGE
  BAL R14,MESSR DISPLAY IT

ALRT1 DS ØH
  CLI MCSCDLIM,Ø REACHED QUEUE LIMIT?
  BZ ALRT2 NO
  LA R1,MSGDLIM GET ERROR MESSAGE
  BAL R14,MESSR DISPLAY IT

ALRT2 DS ØH
  CLI MCSCINTR,Ø INTERNAL ERROR?
  BZ ALRT3 NO
  LA  R1,MSGINTR GET ERROR MESSAGE
  BAL  R14,MESSR DISPLAY IT

ALRT3 DS ØH
  CLI MCSCALRT,Ø REACHED ALERT PERCENT?
  BZ ALRT4 NO
  LA R1,MSGALRT GET ERROR MESSAGE
  BAL R14,MESSR DISPLAY IT

ALRT4 DS ØH
  LA R1,ALRMSG GET ALERT MESSAGE
  BAL R14,MESSR DISPLAY IT
  BAL R14,DEACT DEACTIVATE CONSOLE

* INIENF: INIT. ENF LISTENER *

INIENF DS ØH
  BAKR R14,Ø  SAVE CALLER ENVIRONMENT
  SAC 0 SET PRIMARY MODE FOR MODESET
  SYSSTATE ASCENV=P LET MACROS KNOW PRIMARY MODE
  MODESET MF=(E,SUPØ) SET SUP STATE AND KEY ZERO
  GETMAIN R,LV=ENFEXITL,SP=241
  ST R1,ENFEXITA
  LR R5,R1
  MVC Ø(ENFEXITL,R1),ENFEXIT
  L R1,ENFEXITA
  ENFREQ ACTION=LISTEN, CODE=ENFC41,
EXIT=(R5).
QUAL=WLMQ12,
QMASK=BYTE1,
EOT=YES

MODESET MF=(E,PROB)
PR
ENFC41 EQU 41
WLMQ12 DC X'40000000'

* ———————————————————————————————————*
* - GETSYS: GET SYSNAME                                          - *
* ———————————————————————————————————*

SYSSTATE ASCENV=P          LET MACROS KNOW PRIMARY MODE
GETSYS DS ØH

BAKR R14,Ø              SAVE CALLER ENVIRONMENT

*   PSA — FLCCVT -> CVT — CVTSMCA -> SMCA
*     +++               +++                ++++
*
*                                    SMFID: SMCASID
*
SR R1Ø,R1Ø
USING PSA,R1Ø
L R2,FLCCVT
USING CVTMAP,R2
*
L R5,CVTSMCA
USING SMCABASE,R5

* SMFID
MVC XSYS,=CL8' '
MVC XSYS(Ø4),SMCASID
MVC CNAME(Ø4),SMCASID

* OVERIDE WITH LOGICAL MVS
MVC XSYSL,XSYS
CLC XSYS,=CL8'RCET'
BE ITISN
CLC XSYS,=CL8'ZMVS'
BE ITISN
CLC XSYS,=CL8'AMVS'
BE ITISW
CLC XSYS,=CL8'BMVS'
BE ITISW
B XSYSOK

ITISN EQU *
MVC XSYSL,=CL8'NMVS'
B XSYSOK

ITISW EQU *
MVC XSYSL,=CL8'WMVS'
B XSYSOK

XSYSOK EQU *
PR

* ———————————————————————————————————*
SYSSTATE ASCENV=P LET MACROS KNOW PRIMARY MODE

ACTCON DS ØH
BAKR R14,Ø SAVE CALLER ENVIRONMENT
L R1,CNID SEE IF I HAVE A CONSOLE ACTIVE
LTR R1,R1 ANY ID?
BNZ ISACT YES, DON'T ACTIVATE ANOTHER
MVC CNAME+1(7),=C'WLMCONS'

CHKNM DS ØH CHECK IF CONSOLE ACTIVE USING CONVCON
XC CONV(CONVPLEN),CONV CLEAR CONVCON PARM LIST
MVC CONVACRO,=C'CONV' SET ACRONYM
MVI CONVRSN,CONVRID SET VERSION
OI CONVFLED,CONVFLD SET NAME TO ID CONVERSION
MVC CONVFLED,CNAME SET CONSOLE NAME
OI CONVGFLG,CONVFLG SET NO AREA VERIFICATION
CONVCON CONV CALL CONVCON
LTR R15,R15 CHECK RC
BNZ DOACT BRANCH IF NOT ACTIVE

ISACT DS ØH
LA R1,DIDACT ERROR, CONSOLE ALREADY ACTIVE
BAL R14,MESSR SHOW MESSAGE
B DELCIB DELETE CIB

DOACT DS ØH
LA R1,OPERPRM BUILD OPERPARM DEFAULTS
USING MCSOPPRM,R1 MAP AREA
XC OPERPRM(MCSOPLEN),OPERPRM CLEAR OPERPARM PARM LIST
MVI MCSOAUTH,MCSOMSTR SET MASTER AUTHORITY
MVI MCSOMIG,MCSOMIGY GET A MIGRATION ID
MVC MCSOKEY,=C'EXAMPLE ' SET KEY
MVI MCSOMSFG,MCSOSLST MSCOPE = SYSTEMS LIST
MVI MCSOMISC,MCSOAUTY
MVI MCSOMTP1,MCSOMTJN+MCSOMTSS
LA R2,SYSLST
ST R2,MCSOMSPT STORE POINTER TO SYSTEMS LIST
MVC SYSLST(4),=F'0001' ONE SYSTEM
MVC SYSLST+4(8),XSYS
MODESET MF=(E,SUP) SET SUP STATE TO ACTIVATE CONSOLE
MCSOPER REQUEST=ACTIVATE ACTIVATE THE CONSOLE
NAME=CNAME, ACTIVATE NAME FOUND IN CNAME
TERMNAME=CNAME, USE CNAME FOR THE TERMNAME AUDIT
OPERPARM=OPERPRM, USE MY OPERPARMS IF NONE IN RACF
MSGDLVRY=FIFO, REQUEST FIFO DELIVERY
MSGECB=ECB, ECB TO BE POSTED WHEN MSG IS QUEUED
ALERTECB=ALERT, ECB TO BE POSTED WHEN ALERT OCCURS
MCSCSA=CSA, RETURNED STATUS AREA ADDRESS
MCSCSAA=CSAALET, RETURNED STATUS AREA ALET
CONSID=CNID, RETURNED CONSOLE ID
RTNCODE=RC, SAVE RETURN CODE
RSNCODE=RSN, SAVE REASON CODE
MF=(E,MCSOPPL)
MODESET MF=(E,PROB) BACK TO PROBLEM STATE
ICM R15,15,RC GET RETURN CODE
BNZ ACTERR IF NON-ZERO, PROCESS ERROR
LA R1,NOWACT NOW ACTIVE MESSAGE
BAL R14,MESSR DISPLAY IT
B DELCIB DONE WITH CIB

ACTERR DS ØH
LA R1,BADINI CONSOLE INITIALIZATION ERROR
BAL R14,MESR DISPLAY MESSAGE
B DELCIB DONE WITH CIB

* ———————————————————————————————————*
*- DOCIB: BRANCH ENTERED ASCMODE=PRIMARY -*
*- FUNCTION: PROCESS ALL CIBS QUERIED TO THIS JOB -*
*- OPERATION: -*
*— LOOP WHILE THERE ARE CIBS -*
*— WHEN MODIFY CIB -*
*— SAVE REQUESTING CONSOLE ID -*
*— WHEN ACTIVATE COMMAND -*
*— INVOKE MCSOPER TO ACTIVATE THE CONSOLE -*
*— DISPLAY ACKNOWLEDGEMENT OR ERROR -*
*— WHEN DEACTIVATE COMMAND -*
*— INVOKE MCSOPER TO DEACTIVATE THE CONSOLE -*
*— DISPLAY ACKNOWLEDGEMENT OR ERROR -*
*— WHEN STOP CIB -*
*— SET DONE INDICATOR -*
*— DELETE THE CIB -*
*— RETURN TO CALLER -*
*——————————————————————————————————*

SYSSTATE ASCENV=P LET MACROS KNOW PRIMARY MODE
DOCIB DS ØH
BAKR R14,Ø SAVE CALLER ENVIRONMENT
MVI CMDRSP,1 ISSUE WTOS AS COMMAND RESPONSE
CIBLP DS ØH
ICM R7,15,COMCIBPT GET ADDRESS OF THE CIB
BNZ SVINFO GOT ONE, CHECK THE CIB TYPE
MVI CMDRSP,Ø TURN OFF CMD RESPONSE FLAG
PR NO, RETURN TO CALLER
USING CIB,R7 CIB BASED ON R7
SVINFO DS ØH
LR R1,R7 GET CONSID AND CART FROM CIBX
AH R1,CIBXOFF CIBX=ADDR(CIB)+CIBXOFF
USING CIBX,R1 GET ADDRESSABILITY
MVC MYOPER,CIBXCNID GET CONSOLE ID THAT I WILL TALK TO
MVC MCART,CIBXCART KEEP CART FOR A CMD RESPONSE
DROP R1 DONE WITH CIBX
CLI CIBVERB,CIBMODFY CHECK FOR MODIFY
BNE CKSTOP NO, TRY STOP
CLC     MSGID,=F'Ø'  DO I HAVE A MESSAGE TO BE DOMED  
BE      DOFCMD     NO  
DOM     MSG=MSGID  DOM IT  
XC      MSGID,MSGID  CLEAR HELD MESSAGE ID  

DOFCMD  DS  ØH  
LH     R3,CIBDATLN  GET TEXT LENGTH IN R3  
C      R3,=A(L'CMDACT)  CHECK CMD LENGTH  
BL     NOTACT  TOO SHORT  
CLC     CMDACT(L'CMDACT),CIBDATA  CHECK TEXT  
BNE    NOTACT  NOT ACTIVATE  
BAL    R14,REFRES  
B       DELCIB  
PR  

NOTACT  DS  ØH  
C      R3,=A(L'CMDDACT)  CHECK FOR DEACTIVATE COMMAND  
BL     NOTDACT  BAD LENGTH  
CLC     CMDDACT(L'CMDDACT),CIBDATA  CHECK TEXT  
BNE    NOTDACT  NOT DEACTIVATE COMMAND  
BAL    R14,DEACT  DEACTIVATE CONSOLE  
B       DELCIB  DONE WITH CIB  

NOTDACT  DS  ØH  NOT MY COMMAND, ISSUE IT AS MGCRE  
MODESET  MF=(E,SUPØ)  SUP STATE, KEY Ø FOR SVC34  
LA     R2,CIBDATLN  GET ADDRESS OF LENGTH FIELD  
MGCRE     TEXT=(R2),  POINT TEXT TO CIB DATA LENGTH FIELD  
CONSID=CNID,  ISSUE FROM MY CONSOLE  
CART=MCART,  USE INPUT CART TO CORRELATE RESPONSE  
MF=(E, MGCREPL)  LIST FORM IN MGCREPL  
MODESET  MF=(E,PROB)  BACK TO PROBLEM STATE, KEY  
B       DELCIB  DONE WITH CIB  

CKSTOP  DS  ØH  
CLI     CIBVERB,CIBSTOP  CHECK FOR STOP CIB  
BNE    CKSTRT  NOT STOP EITHER  
MVI     DONE,1  SIGNAL DONE  
MVC    ENDED,=X'11'  
ICM    R1,15,CNID  GET CONSOLE ID  
BZ      DELCIB  THE CONSOLE IS NOT ACTIVE  
BAL    R14,DEACT  DEACTIVATE IT  
B       DELCIB  DONE WITH CIB  

CKSTRT  DS  ØH  
CLI     CIBVERB,CIBSTART  CHECK FOR START CIB  
BNE    DELCIB  CIB NOT USED BY THIS PROG  
MVI     STRTD,1  THIS IS A STARTED TASK  

DELCIB  DS  ØH  
QEDIT ORIGIN=COMCIBPT,  FREE THE CIB  
    BLOCK=(R7)  GO LOOK FOR ANOTHER  

*- DEACT: BRANCH ENTERED, SETS ASCMODE=PRIMARY  -*  
*- FUNCTION: DEACTIVATE THE CONSOLE  -*  
*- OPERATION:  -*  

© 1998. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (940) 455 7050, fax (940) 455 2492.
*- SAVE CALLER STATE -*
*- SET SUP STATE FOR MCSOPER DEACTIVATE -*
*- IF RC IS Ø THEN -*
*- SET CURRENT CONSOLE ID TO Ø (CNID) -*
*- DISPLAY CONSOLE DEACTIVATED MESSAGE -*
*- ELSE -*
*- DISPLAY DEACTIVATION ERROR MESSAGE -*

DEACT DS ØH

BAKR R14,Ø SAVE CALLER STATE
SAC Ø RUNS IN PRIMARY MODE
SYSSTATE ASCENV=P TELL MACROS
MODESET MF=(E,SUP) SET SUP STATE
MCSOPER REQUEST=DEACTIVATE, X
  CONSID=CNID, DEACTIVATE THE CONSOLE X
  RTNCODE=RC, SAVE RETURN CODE X
  RSNCODE=RSN, SAVE REASON CODE X
  MF=(E,MCSOPPL)
MODESET MF=(E,PROB) SET PROBLEM STATE
ICM R15,15,RC GET RETURN CODE
BNZ DACTERR IF NON-ZERO, PROCESS ERROR
XC CNID,CNID ZERO CONSOLE ID TO SHOW NOT ACTIVE
LA R1,NOWDACT GET NOT ACTIVE MESSAGE
BAL R14,MESSR DISPLAY IT
PR RETURN

DACTERR DS ØH
LA R1,BADDACT DEACTIVATE ERROR
BAL R14,MESSR DISPLAY MESSAGE
PR RETURN

*- MESSR: BRANCH ENTERED, R1=ADDR(MESSAGE), SETS ASCMODE=PRIMARY -*
*- FUNCTION: DISPLAY A MESSAGE -*
*- OPERATION: -*
*- DOES A WTO OF THE MESSAGE PASSED AS THE PARAMETER -*

MESSR DS ØH

BAKR R14,Ø SAVE CALLER ENVIRONMENT
SAC Ø RUN IN PRIMARY MODE
SYSSTATE ASCENV=P TELL MACROS PRIMARY MODE
LR R2,R1 USE R2 FOR TEXT IN WTO
CLI CMDRSP,1 CHECK FOR COMMAND RESPONSE
BE MESSRC YES, ISSUE AS CMD RESPONSE
WTO TEXT=(R2), DISPLAY MESSAGE X
  MF=(E,WTOPL)
PR RETURN TO CALLER

MESSRC DS ØH

WTO TEXT=(R2), DISPLAY MESSAGE X
  CONSID=MYOPER, X
  CART=MCART, X
  MF=(E,WTOPLCR)
PR                        RETURN TO CALLER

*-- MESSAGES -------------- *

*  IEF125I DC   CLB'IEF125I'
  IEF4Ø3I DC   CLB'IEF4Ø3I'
*
  INT     DS   ØD
  HH     DC   X'FØFØ'
  MM     DC   X'FØFØ'
  SS     DC   X'FØFØ'
  TH     DC   X'F5CØ'
*
  BADINI  DC   AL2(L'MSGØ)
  MSGØ   DC   C'WJSØ00I ERROR ACTIVATING CONSOLE'
  BADGET  DC   AL2(L'MSG1)
  MSG1   DC   C'WJSØ01I ERROR TRYING TO GET A MESSAGE'
  DIDACT  DC   AL2(L'MSG2)
  MSG2   DC   C'WJSØ02I CONSOLE IS ALREADY ACTIVE'
  NOWACT  DC   AL2(L'MSG3)
  MSG3   DC   C'WJSØ03I CONSOLE HAS BEEN ACTIVATED'
  NOWDACT DC   AL2(L'MSG5)
  MSG5   DC   C'WJSØ05I CONSOLE HAS BEEN DEACTIVATED'
  BADDACT DC   AL2(L'MSG6)
  MSG6   DC   C'WJSØ06I ERROR DEACTIVATING CONSOLE'
  ALRMSG  DC   AL2(L'MSG7)
  MSG7   DC   C'WJSØ07I ALERT DETECTED - DEACTIVATING CONSOLE'
  INITMSG DC   AL2(L'MSG8)
  MSG8   DC   C'WJSØ08I NOW ACCEPTING MODIFY COMMANDS'
  MSGMLIM DC   AL2(L'MSG9)
  MSG9   DC   C'WJSØ09I CONSOLE QUEUEING STOPPED DUE TO MEMORY LIMIT'
  MSGDLIM DC   AL2(L'MSG10)
  MSG10   DC   C'WJSØ10I CONSOLE QUEUEING STOPPED DUE TO DEPTH LIMIT'
  MSGINTR DC   AL2(L'MSG11)
  MSG11   DC   C'WJSØ11I INTERNAL SYSTEM ERROR ON CONSOLE'
  MSGALRT DC   AL2(L'MSG12)
  MSG12   DC   C'WJSØ12I RECEIVED QUEUE DEPTH ALERT'
  INITMS2 DC   AL2(L'MSG13)
  MSG13   DC   C'WJSØ13I WAITING FOR FIRST COMMAND'

*-- STATIC VARIABLES ------- *

  CMDACT  DC   C'REFRESH'         ACTIVATE COMMAND
  CMDDACT DC   C'DEACTIVATE'      DEACTIVATE COMMAND

* WTO TO DEBUG
  WTOC   WTO   'X
             ',MF=L,ROUTCDE=(11)
  WTOL   EQU   *-WTOC               LENGTH OF MACRO EXPANSION
  RECOVERY DS   ØH

© 1998. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (940) 455 7050, fax (940) 455 2492.
DROP .
USING *,R15 SET UP ADDRESSABILITY
*
LA R4,12
CR R0,R4 IS SDWA PRESENT?
BNE HAVESDWA YES, BR TO PROCESS WITH SDWA
*
LA R15,0 SET RETCODE TO PERCOLATE
BR R14 RETURN
HAVESDWA DS 0H ENTER HERE IF SDWA PRESENT
USING SDWA,R1
L R11,SDWAPARM ADDRESS OF WORK AREA
L R11,0(R11)
USING DYNMODEL,R11
L R12,BASE_REG
DROP R15
USING WLMRESET,R12
ST R14,SAVE_R14 SAVE RETURN ADDRESS
ST R12,SDWASR12 BASE REGISTER FOR RETRY
ST R11,SDWASR11 WORK AREA FOR RETRY
ST R13,SDWASR13 WORK AREA FOR RETRY
SETRP RC=4, RETADDR=RETRYPT,
RETREGS=YES,FRESHDWA=YES
*
L R14,SAVE_R14 RESTORE RETURN ADDRESS
BR R14
*
LTORG
ENFEXIT SAVE (14,12)
BASR R12,0 R12 = BASE REGISTER
USING *,R12
GETMAIN R,LV=WORKLE
ST R1,8(R13)
ST R13,4(R1)
LR R13,R1
USING WORKE,R13
LA R2,CMD
MGCRE MF=(E,LAREA),TEXT=(R2),CONSID=MASTER
L R13,4(R13) RESTORE R13
L R1,8(R13)
FREEMAIN R,LV=WORKLE,A=(R1)
L R14,12(R13)
LM R0,R12,20(R13)
SR R15,R15 SET UP RC
BR R14 RETURN TO MVS AND USE RC=R15
CMD DS 0CL6
CMDLEN DC XL2'20'
XCMD DC CL20'F WLMRESET,REFRESH '
TOKENC DS CL1
MASTER DC F'00'
LAREA MGCRE MF=L
ENFEXITL EQU *-ENFEXIT

* - DYNAMIC AREA MODEL -*

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

DYNMODEL DS 0F
REGISTER DS F

ECBS DS 0CL12 ECB LIST FOR WAIT
MECB DS A ADDR(MESSAGE ECB)
AECB DS A ADDR(ALERT ECB)
MODECB DS A ADDR(MODIFY/STOP ECB)

CNID DC F'0' CONSOLE ID

CSA DS A ADDR(MCSCSA)
CSAALET DS F ALET(MCSCSA)

ECB DC F'0' MESSAGE ECB
ALERT DC F'0' ALERT ECB

COMADDR DS F ADDR(COMAREA) FROM EXTRACT
RC DS F RETURN CODE FROM MCSOPER/MCSOPMSG
RSN DS F REASON CODE FROM MCSOPER/MCSOPMSG

MYOPER DS F CONSOLE ID FROM LAST MODIFY COMMAND

ENFPTR DS A
ENFEXITA DS A

MSGID DC F'0'
SYSLST DS 17F MSCOE SYSTEMS LIST

MYRASD DS 0F
DS CL(RASD_LEN)

MYRASC DS 0F
DS CL(RASC_LEN)

OPERPRM DS CL(MCSPLEN) OPERPARMS AREA
MCART DS CL8 CART FROM MESSAGE OR CIB
CNAME DS CL8 CONSOLE NAME TO ACTIVATE
DC CL2' ' SPACE FOR AREA ID ON CONVCON

SV DS 18F SAVE AREA
TOFF DS F OFFSET TO MESSAGE IN TEXT OBJECT

DONE DC FL1'0' DONE FLAG
MDBFLGS DC FL1'0' MDB FLAGS

MDBFGO EQU X'01' PROCESSED GENERAL OBJECT

MDBFCO EQU X'02' PROCESSED CONTROL PROG OBJECT
CMRSP DC FL1'0' COMMAND RESPONSE FLAG

STRTD DC FL1'0' INDICATOR THAT THIS WAS STARTED TASK
DS 0H

WTOBUF DS FL2 LENGTH FOR DYNAMIC MESSAGES
WTOID DS CL8 MESSAGE ID FOR ECHOED MESSAGES

WTOTXT DS CL118 MESSAGE TEXT

XSRV DS CL8
XJOB DS CL8
XSYS DS CL8
XSYSL DS CL8
XASID DS H
ENDED DS X
REG DS F
TCBADDR DS F
EJECT

WTOPL WTO TEXT=, WTO PARAMETER LIST X
DESC=(7), X
MF=L
WTOPLCR WTO TEXT=, WTO PARAMETER LIST FOR CMD RESPONSE X
CONSID=, X
CART=, X
DESC=(5,7), DESCRIPTOR CODE 5 IS CMD RESPONSE X
MF=L
WTOHOLD WTO TEXT=, WTO PARAMETER LIST TO HOLD MSGS X
DESC=(3,7), X
MF=L
EJECT

MGCREPL MGCRE MF=(L) MGCRE PARAMETER LIST
EJECT

SUP MODESET MODE=SUP, MF=L MODESET PARM LIST FOR SUP STATE X
SUPØ MODESET MODE=SUP, KEY=ZERO, MF=L MODESET PARM LIST FOR SUP, KEY Ø X
PROB MODESET MODE=PROB, KEY=NZERO, MF=L MODESET PARM LIST FOR PROBLEM STATE X
EXTRACT EXTRACT MF=L EXTRACT PARAMETER LIST
EJECT
IEZVG2ØØ DSECT=NO CONVCON PARAMETER LIST
EJECT
MCSOPER MF=(L,MCSOPPL) MCSOPER PARAMETER LIST
EJECT
MCSOPMSG MF=(L,MCSOPMPL) MCSOPMSG PARAMETER LIST
WTO DS CL(WTOL)
BASE_REG DS F
SAVE_R14 DS F
DYNL EQU *-DYNMODEL DYNAMIC AREA LENGTH
WTOCE WTO ' ',MF=L, ROUTCDE=(11)
WTOLE EQU *-WTOCE LENGTH OF MACRO EXPANSION
WORKE DSECT
SAVE DS 18F
WTOE DS CL(WTOLE)
WORKLE EQU *-WORKE

*---------------------------------------------------------------*
* REQUIRED DSECTS                                              *
*---------------------------------------------------------------*
EJECT
IEAVG132 , MDB PREFIX
EJECT
IEAVM1Ø5 , MDB
EJECT
IEAVG131 , CONSOLE STATUS AREA

WLMRESET

//WLMRESET PROC
/**
//** TIME=NOLIMIT TO AVOID S522 ABEND
//**
//WLMRESET EXEC PGM=WLMRESET,TIME=NOLIMIT
/**
This procedure is started automatically during IPL using SYS1.PARMLIB(COMMND00). It can be stopped using: PWLMRESET. Load module WLMRESET must be link-edited in an authorized library with AC=1.

LOAD MODULE WLMRESET

//I990557B JOB (01808),
// 'SYSTEM TEAM',
// MSGCLASS=R,
// MSGLEVEL=(1,1),
// NOTIFY=I990557,
// CLASS=4
/*
/*
******************************************************************************
//ASSEM    PROC MEMBER=
//ASSEM    EXEC PGM=IEV90,
//        PARM=('NODECK,OBJECT,NOXREF')
//SYSLIB   DD  DISP=SHR,DSN=SYS1.MODGEN
//        DD  DISP=SHR,DSN=SYS1.MAACLIB
//        DD  DISP=SHR,DSN=SYS1.AMACLIB
//        DD  DISP=SHR,DSN=SYS1.ICEMAC
//SYSUT1   DD  DSN=&SYSUT1,SPACE=(1024,(120,120),..,ROUND),UNIT=SYSALLDA
//SYSPRINT DD  SYSOUT=* 
//SYSLIN   DD  DSN=&OBJ,SPACE=(3040,(40,40),..,ROUND),UNIT=SYSALLDA,
//         DD  DDNAME=SYSIN
//SYSLMOD  DD  DISP=SHR,DSN=SYS1.LINKLIB(&MEMBER)
//SYSUT1   DD  DSN=&SYSUT1,SPACE=(1024,(120,120),..,ROUND),UNIT=SYSALLDA
//SYSPRINT DD  SYSOUT=* 
//PEND
******************************************************************************
//LINK    PROC MEMBER=,PRM=' ' 
//LINK    EXEC PGM=HEWLO96,PARM=''.&PRM'
//SYSLIN   DD  DSN=&OBJ,DISP=(OLD,DELETE)
//        DD  DDNAME=SYSIN
//SYSLMOD  DD  DISP=SHR,DSN=SYS1.LINKLIB(&MEMBER)
//SYSUT1   DD  DSN=&SYSUT1,SPACE=(1024,(120,120),..,ROUND),UNIT=SYSALLDA
//SYSPRINT DD  SYSOUT=* 
//PEND
******************************************************************************
//ASSEM    EXEC PROC=ASSEM,MEMBER=WLMRESET
//LINK    EXEC PROC=LINK,MEMBER=WLMRESET,PRM='AC=1'
******************************************************************************
/*
INTRODUCTION
This program searches partitioned datasets for strings of text and then replaces that string with another. When a specified string is found the string, record, and member are flagged. This flag is only used to create summaries of the changes made for both the member and dataset.

It is expected that the strings to be replaced (targets) and the replacement strings (objects) be refined by using YEAR2K and that most probably the replacement would be made to members extracted to another PDS.

There are a few warnings which apply to any program that makes global changes, especially to all members of a PDS. First is that all programs can have errors (even this author’s) and second is that global requests might have results that were not intended but were made because of replacement coding errors, etc. Therefore, it is suggested that the original PDS be recoverable (eg, by making a back-up, etc) before applying global changes.

SEARCH/REPLACE STRING SPECIFICATIONS
Strings are defined by labels WORDLIST through LASTWORD and the definition is by macro STDEF. One exception is that instead of only the search string a second parameter containing the string to be substituted is also specified. Another exception is that a keyword (OPTION=) is provided and is described below. This macro is defined within the program source and may contain from two to six operands, as follows:

- The character strings (first two operands). These character strings may contain any EBCDIC characters. If embedded blanks, commas, or single quotation marks are included, the string must be contained in single quote marks. If embedded quote marks or ampersands are desired, each occurrence must be specified as two consecutive specifications of that character (ie, ‘’ or && to specify ‘ or &\&, respectively). The first string is the string to be replaced and the second is the replacement string.
• The remaining positional operands, if present, indicate that the search is qualified to specific segment(s) of the specified string. These operands consist of the single characters W, P, and/or S to denote qualifications of WORD, PREFIX, and/or SUFFIX respectively. These qualifiers have the same meaning as those used in ISPF search and replace commands. For example, if word and prefix are specified for string DATE, the strings DATE and DATE2 will be selected, but UPDATE will not be selected. If all three qualifiers are specified for string MM; MM, MMDDYY, and YYMM will qualify, while SUMMARY will not qualify. It should be noted that the program logic first processes, for each record, those entries without these qualifiers and then again those with these qualifiers. This may or may not be desirable, but it is a program ‘feature’ and, hence, the user should be aware of this processing order.

• The keyword ‘OPTION’= is used to indicate the action to take if the target string is longer than the source string and the attempt to remove sufficient spaces is insufficient. Values are FORCE and ABORT (default). In the first case, sufficient non-blank characters at the end of the record are deleted. In the second case, the replacement is not made and a warning message is issued. Before the indicated action is taken, all multiple occurrences of spaces (not enclosed in single quotation marks) are replaced by a single occurrence. See below for more details on this compression technique.

```
WORDLIST DS OC
PUSH PRINT
PRINT GEN
STDEF SPACE,'C'' ','W
STDEF ZERO,LOW-VALUE
STDEF XYZ-DATE,XYZ-NEW-DATE,OPTION=FORCE
STDEF XYZ-YY,XYZ-CCYY
STDEF 'QUOTE''TEST',NEW''QUOTE''TEST
LASTWORD DC X'FF' NOTE THAT THIS MUST IMMEDIATELY X
FOLLOW LIST OF CHARACTER STRINGS
POP PRINT
```

Sample definitions are shown in Figure 1. These strings are not expected to apply to Year 2000 processing but are designed mainly to test different replacement conditions. Note that the last macro must
be immediately followed by a byte of all ones. The PRINT option is included to show the generated entries and may be removed if desired.

MEMBER SELECTION
Members of the PDS may be limited in two ways:

- FROM=member1 and THRU=member2 PARM fields. These specifications limit member names to those from member1 through member2, whose respective default values are the first and last members of the PDS. For example, PARM='FROM=C,THRU=M' would restrict analysis to members beginning with characters C through L and the member M.

- Use of the exclusion dataset (CARDS). Records from this sequential dataset are read, information from bytes 1-8 is extracted and sorted. Member names that match any of these selections are excluded from analysis. If bytes 2-8 contain an asterisk, all members whose names match the previous characters are excluded. For example, the entries MEMBERX and NAME* would exclude the members MEMBERX and all members whose first four characters are NAME.

OTHER EXEC STATEMENT = OPTIONS
In addition to the above, the option ‘PRNT=’ provides specialized printing options. Multiple options may be specified by enclosing in parentheses and separating by commas, eg PRNT=(BEFORE,AFTER). The options are:

- DIAG – this option is used for program diagnosis and is not described fully here. It sets a flag and when subroutine TEST is executed certain registers and other data are displayed.

- LIST – this option lists all processed records whether or not they contain strings that are to be replaced.

- BEFORE – if the LIST option is not specified this option lists the original record that contain strings that are to be replaced. The
image is followed by the identifier ‘<= BEFORE’.

- AFTER, lists records after the character string has been replaced (or where such a replacement was attempted but was not allowed). The image is followed by the identifier ‘<= AFTER’. Note that this listing occurs for each replacement string in the record.

**SAMPLE JCL**

```plaintext
//SYSTØØ2I JOB ...
//STEP1 EXEC PGM=YEAR2KR,PARM='PRNT=(BEFORE,AFTER)'  
//SYSABEND DD SYSOUT=*  
//SYSPRINT DD SYSOUT=*  
//PRINTER DD SYSOUT=*  
//ERRORS DD SYSOUT=*  
//PDS DD DSN=YEAR2K.TEST.PDS,DISP=SHR  
//CARDS DD *  
L8Ø*  
```  

Sample JCL for executing the program is provided above. Samples of the processed records (before and after replacements are made) are provided in Figure 3.

**ERROR CONDITIONS AND REPORT**

If the replacement is longer than the replaced string, the statement is analysed to see if space is available between its fields or at the end of the statement. If insufficient space is found, the replacement is made by removing characters from the end of the statement. In this case the before and after images are written to report ERRORS and a message is appended to the normal output report (PRINTER).

**RESTRICTIONS AND SPECIAL CONDITIONS**

This program is designed to process 80-byte records of which the last eight bytes are not subject to change. As noted above, replacement of strings that may be embedded within other strings is processed first. After that, the record is reprocessed with those strings that are defined with word, prefix, or suffix qualifiers.
Before replacements:

<table>
<thead>
<tr>
<th>Date</th>
<th>Details</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>05-10-15</td>
<td>This is an example of no compression XYZ-YY</td>
<td>123456789712</td>
</tr>
<tr>
<td>05-10-20</td>
<td>This is an example of left compression XYZ-YY X</td>
<td>123456789712</td>
</tr>
<tr>
<td>05-10-25</td>
<td>This is an example of right compression XYZ-YY XXX XXX</td>
<td>123456789712</td>
</tr>
<tr>
<td>05-10-30</td>
<td>'RESERVED ' above with quoted string XYZ-YY XXX XXX</td>
<td>123456789712</td>
</tr>
<tr>
<td>05-10-35</td>
<td>This is an example of left and right compression XYZ-YY XXX XXXX</td>
<td>3456789712</td>
</tr>
<tr>
<td>05-10-40</td>
<td>This is an example with imbedded string IMXYZ-YYBED</td>
<td>123456789712</td>
</tr>
<tr>
<td>05-10-45</td>
<td>This is an example of no space xyz-yy ............</td>
<td>123456789712</td>
</tr>
<tr>
<td>05-10-50</td>
<td>This is an example of option=force ... xyz-date ........</td>
<td>123456789712</td>
</tr>
<tr>
<td>05-10-55</td>
<td>After replacements:</td>
<td></td>
</tr>
<tr>
<td>05-10-60</td>
<td>Date</td>
<td></td>
</tr>
<tr>
<td>05-10-65</td>
<td>This is an example of no compression XYZ-CCYY</td>
<td>123456789712</td>
</tr>
<tr>
<td>05-10-70</td>
<td>This is an example of left compression XYZ-CCYY X</td>
<td>123456789712</td>
</tr>
<tr>
<td>05-10-75</td>
<td>This is an example of right compression XYZ-CCYY XXX XXXX</td>
<td>123456789712</td>
</tr>
<tr>
<td>05-10-80</td>
<td>'RESERVED ' above with quoted string XYZ-CCYY XXX XXXX</td>
<td>123456789712</td>
</tr>
<tr>
<td>05-10-85</td>
<td>This is an example of left and right compression XYZ-CCYY XXX XXXX</td>
<td>3456789712</td>
</tr>
<tr>
<td>05-10-90</td>
<td>This is an example with imbedded string IMXYZ-CCYYBED</td>
<td>123456789712</td>
</tr>
<tr>
<td>05-10-95</td>
<td>This is an example of no c' ' ........ xyz-yy ............</td>
<td>123456789712</td>
</tr>
<tr>
<td>05-10-100</td>
<td>This is an example of option=FORCE ... xyz-new-date ........</td>
<td>123456789712</td>
</tr>
</tbody>
</table>

*Figure 1: Sample records before/after processing*

In testing an unintended result was noticed. This provides a good example of some special conditions. The intent was to replace the string ‘SPACE’ by the string ‘C’ ‘’. The results emphasize the importance of carefully considering the effects of string specifications, the demonstration of the above processing order, and an example of multiple listing of ‘AFTER’ records. Further, it may be noted that this processing resulted in the successful replacement of a string in a record where a previous attempt failed. This first action resulted in the logging of an error for insufficient space the second did not. This resulted in the error log ‘AFTER’ record showing no change yet the record was indeed changed by this second attempt.

REPLACEMENT AND COMPRESSION RULES

The following rules are for replacement and compression:
• If the target (string to be replaced) is longer than the object (replacement string) and the target is followed by a space, then the replacement is made and the result is padded with spaces to overlay the right portion of the target that was not replaced by the object.

• If the target is longer than the object and the target is not followed by a space, then the replacement is made, the remainder of the image (first 72 bytes of the record) is moved left by the difference in the lengths of the target and object, and the vacated bytes are padded with spaces.

• If the target and object are the same length, the target is replaced by the object.

• If the object is longer than the target, the record is scanned for consecutive spaces and each occurrence is replaced by a single space (until sufficient space is obtained) and, if sufficient space is obtained, the replacement is made. The first record position is never overlaid.

• In the above case if sufficient space is not recovered, replacement is either FORCEd or ABORTed depending on the respective value of the ‘OPTION=’ parameter. If OPTION=FORCE, bytes from the right of the record are removed until sufficient space is obtained and the replacement is made after shifting the contents from the target to the end of the record. If OPTION=ABORT is specified, no replacement is made. In either case warning records (before and after images) are written to the error log. Note that ABORT terminates the update of the individual record and not the execution of the program.

The above does not yield exactly the same results as the equivalent ISPF CHANGE commands (ie spacing may be different), but is designed to minimize replacement failures and is intended to provide, in general, a more desirable source statement structure for at least some languages.

PROGRAM SOURCE

```plaintext
GBLA &N,&IMBED,&OTHER,&WORD,&PREFIX,&SUFFIX
```
LCLC &MYNAME
&MYNAME SETC 'YEAR2KR' CSEC NAME
RBASE EQU 12 BASE REGISTER FOR CSECT
RBAL EQU 10 BAL REGISTER
TITLE '&MYNAME' LISTING TITLE

***********************************************************************
***   THIS PROGRAM SEARCHES PDS SOURCE MEMBERS FOR SPECIFIED       ***
***   CHARACTER STRINGS (TARGETS) AND REPLACES THEM WITH            ***
***   DESIGNATED REPLACEMENT STRINGS (OBJECTS).                    ***
***                                                                 ***
***   IF THE OBJECT STRING IS LARGER THAN THE STRING IT IS TO        ***
***   REPLACE AN ATTEMPT IS MADE TO COMPRESS BLANKS FROM THE        ***
***   ORIGINAL TEXT. IF THAT IS INSUFFICIENT SPACE, A DECISION      ***
***   IS MADE TO EITHER DROP CHARACTERS FROM THE RIGHT OF THE       ***
***   STATEMENT OR TO ELIMINATE THE REPLACEMENT BY THE PARAMETER    ***
***   OPTION=FORCE OR OPTION=ABORT, RESPECTIVELY. THE DEFAULT IS     ***
***   OPTION=ABORT.                                               ***
***                                                                 ***
***   THE CHARACTER STRINGS ARE FOUND IN THE TABLE DEFINED AT LABEL  ***
***   'WORDLIST' BY THE MACRO 'STDEF'.                            ***
***                                                                 ***
***   IT SHOULD BE NOTED THAT THE PROGRAM LOGIC IS TO FIRST          ***
***   PROCESS THE IMBEDDED CHARACTERS STRINGS (I.E., THOSE NOT       ***
***   DEFINED WITH WORD, PREFIX, AND/OR SUFFIX QUALIFIERS), HENCE    ***
***   WHEN THE SCAN FOR THESE QUALIFIED STRINGS ARE PROCESSED IT     ***
***   WOULD BE POSSIBLE TO REPROCESS THE MODIFIED CHARACTER STRING.  ***
***********************************************************************

EJECT

***********************************************************************
***      LINKAGE CONVENTIONS ENTERING PROGRAM                       ***
***********************************************************************

MACRO
&NAME STDEF &A,&X,&B,&C,&D,&OPTION=ABORT
GBLA &N,&IMBED,&OTHER,&WORD,&PREFIX,&SUFFIX
LCLA &K,&F,&L,&I,&J
LCLC &T,&Z
&T SETC '&A'
&I SETA 1
&J SETA 0
&K SETA K'&A
AIF ('&A'(1,1) NE ''').NOTQ
&K SETA &K-2
&T SETC '&A'(2,&K)
.NOTQ AIF (&K GT 0).NOTNULL
MNOTE 8,'NULL TARGET STRING NOT ALLOWED'
MEXIT
.NOTNULL AIF ('&T'(&I,2) NE '*****').NOTDQ
&I SETA &I+1
&J SETA &J+1
.NOTDQ ANOP
&I SETA &I+1

© 1998. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (940) 455 7050, fax (940) 455 2492.
AIF (&I LT &K).NOTNULL
&K SETA &K-&J
&I SETA 1
&J SETA Ø
&Z SETC '&X'
&L SETA K'&X
AIF ('&X'(1,1) NE '***').NOTOX
&L SETA &L-2
&Z SETC '&X'(2,&L)
.NOTQX AIF (&L GT Ø).NOTNULL
MNOTE 8,'NULL OBJECT STRING NOT ALLOWED'
MEXIT
.NOTULX AIF ('&Z'(&I,2) NE '*****').NOTDQX
&I SETA &I+1
&J SETA &J+1
.NOTDQX ANOP
&I SETA &I+1
AIF (&I LT &L).NOTNULL
&L SETA &L-&J
AIF ('&B' NE 'P' AND '&C' NE 'P' AND '&D' NE 'P').NOTP
&F SETA &F+&PREFIX
.NOTP AIF ('&B' NE 'S' AND '&C' NE 'S' AND '&D' NE 'S').NOTS
&F SETA &F+&SUFFIX
.NOTS AIF ('&B' NE 'W' AND '&C' NE 'W' AND '&D' NE 'W').NOTW
&F SETA &F+%&WORD
.NOTW AIF ('(&OPTION' EQ 'FORCE').NOTR
&F SETA &F+X'BØ'
.NOTR ANOP
&NAME DC AL1(&K-1,&L-1,&F),CL&K'T',CL&L'&Z'
&N SETA &N+1
AIF (N'&SYSLIST LE 2).IMBED
&OTHER SETA 1
MEXIT
.IMBED ANOP
&IMBED SETA 1
MEND
MACRO
&LABEL SMUMØØ2 &DSECT=YES,&C=Ø
PUSH PRINT
PRINT GEN
*********************************************************************
* MACRO TO DESCRIBE PDS BLDL ENTRY WITH ISPF STATISTICS, ***
* TO BE USED BY 'BLDL' MACRO. ***
* DSECT=YES WILL CAUSE A DSECT TO BE CREATED. ***
* DSECT=NO DATA WILL BEGIN ON A DOUBLEWORD BOUNDRY. ***
* C=_ LABELS WILL BE GU_2XX (_ MAY BE ANY ALPHAMERIC ***
* CHARACTER(S), INTENDED FOR GENERATING MULTIPLE ***
* COPIES OF THE GENERATED LAYOUT). ***
* *** THIS MACRO IS A MODIFICATION TO 'GTEUMØ2' FROM THE ***
*** CONNECTICUT BANK TAPE.  THE IMPLEMENTATION OF THIS SOURCE ***
*** MANAGEMENT SYSTEM WAS MUCH EASIER BY UTILIZING THIS EXISTING ***
*** CODE.  MUCH GRADITUDE AND APPRECIATION IS GIVEN TO: ***
*
*  CHUCK HOFFMAN, SYSTEMS PROGRAMMING, GTEL COMPUTING CENTER  ***
*
*  MODIFICATION OF HIS MACRO ON THE CONNECTICUT BANK TAPE EASED  ***
*  THE IMPLEMENTATION OF THIS SYSTEM.  ***

*********************************************************************
AIF ('&DSECT' EQ 'YES').GUMØ2A
&LABEL DS ØD  , ISPF STATS PDS BLDL ENTRY
AGO .GUMØ2B
.GUMØ2A ANOP
&LABEL DSECT  , ISPF STATS PDS BLDL ENTRY
.GUMØ2B ANOP
GU&C.2FF DS XL2  BLDL COUNT OF ENTRIES
GU&C.2LL DS XL2  BLDL LENGTH OF ENTRIES
GU&C.2NAM DS CL8  MEMBER NAME
GU&C.2TTR DS XL3  PDS VALUE 'TTR'
GU&C.2K DS X  BLDL VALUE 'K'
GU&C.2Z DS X  BLDL VALUE 'Z'
GU&C.2C DS X  PDS VALUE 'C'
GU&C.2VER DS X  ISPF VERSION NUMBER (BIN)
GU&C.2MOD DS X  ISPF MOD NUMBER (BIN)
DS XL2  (UNUSED, X'0000')
GU&C.2DATC DS PL4  ISPF DATE CREATED (PACK)
GU&C.2DATM DS PL4  ISPF DATE MODIFIED (PACK)
GU&C.2TIMM DS XL2  ISPF TIME MODIFIED (PK NOSIGN)
GU&C.2SIZE DS XL2  ISPF SIZE (BIN)
GU&C.2INIT DS XL2  ISPF INITIAL SIZE (BIN)
GU&C.2MODL DS XL2  ISPF COUNT OF MOD LINES (BIN)
GU&C.2ID DS CL7  ISPF USERID
DS CL3  (UNUSED X'404040')
POP PRINT
MEND

&MYNAME CSECT .
STM R14,R12,12(R13)  SAVE REGS TO CALLER S.A.
B (BEGIN-&MYNAME)(R15)  BRANCH AROUND EYECATCHER
DC A(L'NAME)  LENGTH OF CSECT NAME
NAME DC C'&MYNAME'  CSECT NAME
DC C' &SYSDATE &SYSTIME '  ASSEMBLY DATE/TIME STAMP
BEGIN LR RBASE,R15  LOAD BASE REGISTER
USING &MYNAME,RBASE  ADDRESSABILITY
PRINT NOGEN
GETMAIN R,LV=WORKDLEN  GET SAVE/WORK AREA
ST R1,8(Ø,R13)  MY S.A. ADDR INTO CALLER S.A.
ST R13,4(Ø,R1)  CALLER S.A. ADDR INTO MY S.A.
LR R13,R1  R13 POINTS TO MY S.A.
USING WORKD,R13  ADDRESSABILITY OF SAVE AREA
L R1,4(Ø,R13)  R1 POINTS TO CALLER S.A.
LM R15,R1,16(R1)  R15 RB AND R1 ARE RESTORED
EJECT

***********************************************************************
***      MAINLINE ROUTINE                                           ***
***********************************************************************
MAIN    EQU   *            BEGIN MAINLINE ROUTINE
ST    R1,R1SAVE           SAVE INITIAL R1
XC   COMPCODE,COMPCODE     CLEAR COMPLETION CODE
L   R1,=A(INITIAL)        POINT TO INITIALIZATION ROUTINE
BALR   RBAL,R1            GO PERFORM INITIALIZATION
MAINDIRL   BAL   RBAL,GETDIR   GET MEMBER NAME
LTR   R15,R15             END OF DIRECTORY REACHED?
BNZ   MAINEND             YES
MVI   SWITCHES,Ø            CLEAR ALL CONDITION FLAGS
ZAP   CARDS,=P'Ø'          INITIALIZE RECORD COUNT
L   R3,EXCLUDE1           POINT TO CURRENT EXCLUSION
LR    R4,R3               POINT TO BEGINNING OF MEMBER NAME
LA   RØ,7                 MAXIMUM LENGTH-1
MAINWC  CLI   1(R4),C'**'    WILD CARD PATTERN?
BE   MAINWCX              YES
LA   R4,1(R4)             POINT TO NEXT CHARACTER
BCT   RØ,MAINWCX          CONTINUE
MAINWCX   SR   R4,R3        GET LENGTH-1
MAINXL  EX   R4,MAINXCLC   IS MEMBER TO BE EXCLUDED?
BL   MAINNX              NO
BH   MAINXMB              MAYBE
AP   EXCLUDED,=P'1'        COUNT EXCLUSION
MVC   LINE+9(8),MEMBER     MOVE MEMBER NAME TO OUTPUT LINE
MVC   LINE+18(8),=C'EXCLUDED' SET EXCLUSION MESSAGE
MVC   LINE+26(6),EDITPAT   SET EDIT PATTERN
ED   LINE+26(6),EXCLUDED   FORMAT EXCLUSION COUNT
MVI   LINE,C'Ø'            SET TO DOUBLE SPACE
BAL   RBAL,DOUBLESP        ALLOW FOR DOUBLE SPACE
BAL   RBAL,PRINT           GO PRINT LINE
B   MAINDIRL              GO GET NEXT MEMBER
MAINXCLC  CLC   MEMBER(*-*),Ø(R3) IS MEMBER TO BE EXCLUDED?
MAINXMB   LA   R3,L'EXCLUDES(R3) POINT TO NEXT ENTRY
ST   R3,EXCLUDE1          SAVE POSITION
B   MAINXL              GO CHECK
MAINX   ST   R15,INRECLOC  INITIALIZE FOR GETREC
MAINXTR   BAL   RBAL,GETREC  READ RECORD FROM CURRENT MEMBER
AP   CARDS,=P'1'          COUNT CARD IMAGE
LTR   R15,R15             END OF MEMBER REACHED?
BNZ   MAINNOL              NO
MVC   MEMBNAME,MEMBER     MOVE MEMBER NAME
MVC   MEMBERNO,EDITPAT    MOVE EDIT PATTERN
ED   MEMBERNO,MEMBERS+1   FORMAT MEMBER NUMBER
MVC   INAREA,Ø(R1)        MOVE RECORD
MVC   CARDNO,EDITPAT      MOVE EDIT PATTERN
ED   CARDNO,RECORDS+1     FORMAT CARD NUMBER
TM   OPTIONS,LISTBIT      IS LIST REQUESTED?
MVC LINE+1(INAREA+L'INAREA-MEMBNAME'),MEMBNAME NAME,#,IMAGE
BAL RBAL,PRINT PRINT RECORD IMAGE (BEFORE)

MAINNOL MVI HIT,Ø CLEAR 'FIND' FLAG
NI OPTIONS,X'FF'-ALRODYBIT CLEAR RECORD LISTED FLAG
CLI IMDEF,Ø ANY IMBEDDED DEFINITIONA?
BE MAINNOIM NO
BAL RBAL,SCAN1 SCAN FOR IMBEDDED ENTRIES

MAINNOIM CLI OTDEF,Ø ANY NON-IMBED DEFINITIONA?
BE MAINNOOT NO
BAL RBAL,SCAN2 SCAN FOR WORDS, PREFIXES, & SUFFIXES

MAINNOOT TM SWITCHES,UPDATBIT RECORD MODIFIED?
BZ MAINNXTR NO
AP FINDS,=P'1' COUNT RECORD CONTAINING OCCURRENCE(S)
NI SWITCHES,X'FF'-UPDATBIT TURN OFF UPDATE BIT
BAL RBAL,WRITEREC UPDATE RECORD
B MAINNXTR GO GET NEXT RECORD

MAINEND DS ØH
BAL RBAL,HEADPAGE PUT TOTALS ON NEW PAGE
MVC LINE+5(6),EDITPAT SET EDIT PATTERN
ED LINE+5(6),MEMBERS FORMAT MEMBER NUMBER
MVC LINE+12(13),=C'MEMBERS FOUND'
BAL RBAL,PRINT PRINT TOTAL
MVC LINE+5(6),EDITPAT SET EDIT PATTERN
ED LINE+5(6),EXCLUDED FORMAT MEMBER NUMBER
MVC LINE+12(16),=C'MEMBERS EXCLUDED'
BAL RBAL,PRINT PRINT TOTAL
MVC LINE+5(6),EDITPAT SET EDIT PATTERN
SP MEMBERS,EXCLUDED COMPUTE REMAINDER
ED LINE+5(6),MEMBERS FORMAT MEMBER NUMBER
MVC LINE+12(16),=C'MEMBERS ANALYZED'
BAL RBAL,PRINT PRINT TOTAL
MVC LINE+5(6),EDITPAT SET EDIT PATTERN
ED LINE+5(6),MODIFIED FORMAT MEMBERS MODIFIED
MVC LINE+12(16),=C'MEMBERS SELECTED'
BAL RBAL,PRINT PRINT TOTAL
MVI LINE,C'Ø' SET TO DOUBLE SPACE
BAL RBAL,DOUBLESP ALLOW FOR DOUBLE SPACE
MVC LINE+1(1Ø),OCCUR1 SET EDIT PATTERN
ED LINE+1(1Ø),TRECS FORMAT TOTAL RECORD COUNT
MVC LINE+12(16),=C'RECORDS ANALYZED'
BAL RBAL,PRINT PRINT TOTAL
MVC LINE+1(1Ø),OCCUR1 SET EDIT PATTERN
ED LINE+1(1Ø),TFINDS FORMAT TOTAL RECORDS SELECTED
MVC LINE+12(16),=C'RECORDS SELECTED'
BAL RBAL,PRINT PRINT TOTAL
MVC LINE+1(1Ø),OCCUR1 SET EDIT PATTERN
ED LINE+1(1Ø),TSTRINGS FORMAT TOTAL RECORDS SELECTED
MVC LINE+12(17),=C'OCCURRENCES FOUND'
BAL RBAL,PRINT PRINT TOTAL
CP ERRORTOT,=P'Ø' ANY ERRORS?
BNH MAINNONE NO
MVC LINE(2),=C'Ø*'      SET DOUBLE SPACE/SEED
MVC LINE+3(L'LINE-3),LINE+1 SET '* '*...
BAL RBAL,PRINT PRINT FLAG
MVC LINE(40),=C'Ø*** WARNING ***: SEE 'ERRORS' FILE FOR'
MVC LINE+40(L'EDITPAT),EDITPAT SET EDIT PATTERN
ED LINE+40(L'EDITPAT),ERRORTOT FORMAT COUNT
MVC LINE+41+L'EDITPAT(16),=C'POSSIBLE ERRORS.'
BAL RBAL,PRINT PRINT FLAG

MAINNONE DS ØH
* BEGIN DCB CLOSE
CLOSE (PRINTER),MF=(E,PRCLOSL) CLOSE IT
CLOSE (PDSDIR),MF=(E,DRCLOSL) CLOSE PDSDIR
CLOSE (PDS),MF=(E,PDCLOSL) CLOSE PDS
CLOSE (ERRORS),MF=(E,ERCLOSL) CLOSE ERRORS
* END DCB CLOSE
ENDØØ LA R15,Ø SET COMPLETION CODE ØØ
ST R15,COMPCODE INTO STORAGE
B ENDING GO TO ENDING
EJECT
**********************************************************************
***      LINKAGE CONVENTIONS EXITING PROGRAM                        ***
**********************************************************************
ENDING L R14,COMPCODE R14 SAVES COMP CODE
LR R1,R13 R1 SAVES ADDR OF MY S.A.
L R13,4(R1) R13 RESTORED, PTR CALLER S.A.
FREEMAIN R,LV=WORKDLEN,A=(R1) FREE MY SAVE/WORK AREA
LR R15,R14 R15 SET TO COMP CODE
LM R0,R12,20(R13) R0-R12 RESTORED
L R14,12(R13) R14 RESTORED
MVI 12(R13),X'FF' SET COMPLETION SIGNAL
BR R14 RETURN TO CALLER
* BEGIN STUB DEFINE
EJECT
**********************************************************************
***   GET DIRECTORY RECORD                                          ***
**********************************************************************
GETDIR ST RBAL,SAVGDBAL SAVE LINKAGE REGISTER
CLI DFLAG,Ø FIRST TIME?
* BNE GDNOTIST NO
MVI DFLAG,X'FF' SET FLAG
GDRD BAL RBAL,READDIR READ DIRECTORY RECORD
LTR R15,R15 NORMAL RETURN?
* BNZ GDRETURN NO
BNZ GDEND NO
GDNOTIST L R2,DIRENTRY LOAD ADDRESS OF MEMBER DATA
AP TRECS,RECORDS ACCUMULATE TOTAL RECORDS PROCESSED
ZAP RECORDS,=P'Ø' CLEAR MEMBER RECORD COUNT
AP MEMBERS,=P'1' COUNT NUMBER OF MEMBERS
CLI Ø(R2),X'FF' END OF DIRECTORY BLOCK?
BE GDORD YES
MVC MEMBER,Ø(R2) MOVE MEMBER NAME TO OUTPUT AREA
XR    R15,R15       SET NORMAL RETURN
GDRETURN L    RBAL,SAVGBAL   RESTORE LINKAGE REGISTER
BR    RBAL       RETURN
GDEND LA    R15,4       SET END-OF-DIRECTORY EXIT
B    GDRETURN   GO EXIT
EJECT
***********************************************************************
***   READ DIRECTORY RECORD                                         ***
***********************************************************************
READDIR ST    RBAL,SAVRDBAL   SAVE LINKAGE REGISTER
  L    R6,DIRENTRY       LOAD ADDRESS OF CURRENT LOCATION
  LTR   R6,R6           FIRST DIRECTORY BLOCK?
  BZ    RDNXTDIR         YES
  MVI   LINE,C'Ø'       SET TO DOUBLE SPACE
  BAL   RBAL.DOUBLESP    ALLOW FOR DOUBLE SPACE
  MVC   LINE+1(6),EDITPAT SET EDIT PATTERN
  ED    LINE+1(6),MEMBERS FORMAT MEMBER NUMBER
  MVC   LINE+9(8),MEMBER MOVE MEMBER NAME TO OUTPUT LINE
  MVC   LINE+18(LOCCURS),OCCURS
  ED    LINE+18+OCCURS(L'OCCURS),RECORDS FORMAT RECORDS
  ED    LINE+18+OCCURS(L'OCCURS),FINDS " RECORDS FOUND
  ED    LINE+18+OCCURS(L'OCCURS),STRINGS " STRING OCCURS
  BAL   RBAL,PRINT       PRINT MEMBER HEADING LINE
  CP    FINDS,=P'Ø'      ANY FINDS?
  BZ    RDNOTSTAT        NO
  BAL   RBAL,GETSTATS    GET MEMBER STATISTICS
  LTR   R15,R15         STATS OKAY?
  BNZ   RDNOTSTAT        NO
  BAL   RBAL,PUTSTATS    PRINT MEMBER/STATS
  RDNOSTAT AP    TFINDS,FINDS  ACCUMULATE GRAND TOTAL OF SLCTD RECS
  AP    TSTRINGS,STRINGS ACCUMULATE GRAND TOTAL OF STRINGS
  ZAP   FINDS,=P'Ø'      RESET COUNTER (RECORDS FOUND)
  ZAP   STRINGS,=P'Ø'    RESET COUNTER (STRINGS)
  AP    MODIFIED,=P'1'   COUNT MEMBERS MODIFIED
  B    RDNXTDIR         GO GET NEXT ENTRY
RDNXTDIR GET   PDSDIR,DIRBLOCK     READ DIRECTORY RECORD
  LA    R6,DIRBLOCK+2    POINT TO ENTRY
  ST    R6,DIRENTRY     SAVE ADDRESS (NOT REALLY NEEDED)
  LH    R5,DIRBLOCK     LOAD NUMBER NUMBER OF BYTES USED
  STH   R5,DIRSPACE     SAVE
  SH    R5,=H'2'        REDUCE BY LENGTH OF FIELD
  BNP   RDNXTDIR      IF EMPTY DIRECTORY BLOCK, GO TO NEXT
  B    RD1STMEM       GO PROCESS FIRST ENTRY IN BLOCK
RDNXTMEM L    R6,DIRENTRY     LOAD ADDRESS OF CURRENT LOCATION
  LH    R5,DIRSPACE     LOAD REMAINING SPACE IN BLOCK
  IC    R1,11(R6)      LOAC 'C' FIELD
  N    R1,=F'31'       GET USER AREA HALFWORDS (5 LOW BITS)
  LA    R1,12(R1,R1)   BYTES + MEMBER NAME, 'TTR', AND 'C'
  SR    R5,R1         DEDUCT CURRENT ENTRY LENGTH
  AR    R6,R1         POINT TO NEXT ENTRY
RD1STMEM CLI    Ø(R6),X'FF'   LAST DIRECTRY ENTRY?
Editor’s note: this article will be continued next month when the rest of the code will be published.

Keith H Nicaise  
Technical Services Manager  
Touro Infirmary (USA)  
© Xephon 1998
Neon Systems has announced the beta release of Solution Pack for IMS. The products include DB24X7, Dynamic Index Utility, SPEED Loader, and SPEED Unloader. These provide high levels of IMS availability, data integrity, recoverability, and performance.

For further information contact:
Neon Systems Inc, 14141 Southwest Freeway, Suite 6200, Sugar Land, TX 77478, USA.
Tel: (281) 491 4200
Fax: (281) 242 3880 or
Neon Systems UK Ltd, Third Floor, Sovereign House, 26-30 London Road, Twickenham, Middlesex, TW1 3RW, UK.
Tel: (0181) 607 9911
Fax: (0181) 607 9933.

IBM has announced the release of DFSMS Optimizer Version 1 Release 2 for OS/390 and MVS/ESA. Release 2 has enhancements to the HSM Monitor/Tuner (HMT) and the Optimizer Charting Facility. Operational stability has been improved in HMT by separating the monitoring/tuning activity from the workstation communications and moving it into its own address space. This allows monitoring/tuning to run continuously without the need for workstation connections. Improved event recording is incorporated in Release 2 to gather information from HSM as it occurs.

Contact your local IBM representative for further information.

Syncsort has announced Release 2 of its FilePort automated tool for translating data during mainframe to Unix migration projects and for preparing data for data warehouses. In the new release there are versions for mainframe to Unix translation, vice versa, or both. The product outputs converted records on tape, in a file, or through standard input, and data is converted according to its type and the machine for which it is targeted.

For further information contact:
Syncsort Inc, 3958 Ince Boulevard, Culver City, CA 90232, USA.
Tel: (310) 842 9203
Fax: (310) 842 9014 or
Syncsort Ltd, 60 Churchill Square, Kings Hill, West Malling, Kent, ME19 4DU, UK.
Tel: (01732) 849000
Fax: (01732) 875215.

Xephon will be holding its MVS Update ’98 conference at the Chelsea Hotel in London 16-17 June 1998. MVS Update ’98 is designed specifically for technical managers, systems programmers, strategic planners, and other system specialists at MVS/ESA and OS/390 installations, and provides a thorough analysis of new facilities and products in the MVS world, and a full update on the latest technical hints and tips for MVS administrators.

The attendance fee for MVS Update subscribers is £540 plus £63.00 VAT. For further information, please telephone Angela Scott on (01635) 33598.