148 MVS update

January 1999

In this issue

© Xephon plc 1999

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

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

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

Editor
Jaime Kaminski

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

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

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

© Xephon plc 1999. 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.

Using ‘DDNAME=’ in JCL

INTRODUCTION

Recently one of our application developers rang me to say that he had problems running a particular job. The reason for the problem turned out to be sufficiently curious, and potentially dangerous, for me to feel it was worth passing on the information to a wider audience. In order to explain the problem consider the following two JCL decks.

FIRST JOB

```
//JOBA     JOB standard jobcard
//A EXEC   PGM=IEBGENER
//SYSPRINT DD SYSOUT=B
//SYSUT1   DDNAME=INDIRECT
//SYSUT2   DD SYSOUT=*
//INDIRECT DD DSN=first.dataset,DISP=SHR
//         DD DSN=second.dataset,DISP=SHR
//SYSIN    DD DUMMY
```

SECOND JOB

```
//JOBB     JOB standard jobcard
//A EXEC   PGM=IEBGENER
//SYSPRINT DD SYSOUT=B
//SYSUT1   DDNAME=INDIRECT
//INDIRECT DD DSN=first.dataset,DISP=SHR
//         DD DSN=second.dataset,DISP=SHR
//SYSUT2   DD SYSOUT=*
//SYSIN    DD DUMMY
```

THE PROBLEM

Many users would expect the same results from both jobs. However, what actually happens is that both jobs run, but the first job only copies the first dataset, and a warning message IEF694I is issued. In the case of my user, there was so much JCL and other messages that he did not notice this message. As far as he was concerned the job had completed code 0, but he could not understand why only a portion of the data had been used.

© 1999. Reproduction prohibited. Please inform Xephon of any infringement.

CONCLUSIONS

The explanation for the apparent anomaly is that a 'DDNAME=' only redirects the allocation to the first dataset in a concatenation. Any other datasets concatenated to the target of the DDNAME= are concatenated to the DD preceding the 'DDNAME=' target.

In the first example shown above, the first job above second.dataset is concatenated to SYSUT2! In the second example there is no DD between the 'DDNAME=' and the target of the 'DDNAME='. Therefore, in this case, the additional dataset is concatenated to SYSUT1, and the job works. The problem is therefore twofold:

- Not all the data is read.
- Another random DD unexpectedly gets a concatenation and therefore potentially processes more than expected.

If you require more information on this problem I would suggest reviewing APAR 0W32295.

© Xephon 1999

Dynamic LINKLIST

INTRODUCTION

With the introduction of OS/390 1.3.0, you may have noticed that software such as SYSVIEW or OMEGAMON can no longer dynamically update the LINKLIST. You have to use the MVS function provided in the PROGxx member of SYS1.PARMLIB.

THE PROGXX MEMBER

To use MVS dynamic LINKLIST, you must convert the LNKLSTxx into a PROGxx member containing LNKLST statements. You may use multiple members by specifying PROG=(01,02,03) in IEASYSxx. You must remove the LNK=xx statement, otherwise you will get a CSV487I message during IPL, and MVS will use the PROGxx definitions. The PROGxx member can contain other statements such as APF, EXIT, and SYSLIB.

 © 1999. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (940) 455 7050, fax (940) 455 2492.

IBM provides an edit macro, named CSVLNKPR, in SYS1.SAMPLIB, that converts a valid LNKLSTxx into PROGxx syntax. Once converted, your PROGxx member may look like this:

```
LNKLST DEFINE NAME(linkset.ipl)                                  (1)
LNKLST ADD NAME(linkset.ipl) DSNAME(ANF.SANFLOAD)                (2)
. . .   .    .

LNKLST ADD NAME(linkset.ipl) DSNAME(TCPIP.SEZALNK2)
LNKLST ACTIVATE NAME(linkset.ipl) /* DO NOT SUPPRESS */         (3)
```

1 This statement defines the LNKLST SET named linkset.ipl, which will be activated during IPL. A LNKLST SET consists of an ordered list of datasets for processing as the LNKLST concatenation. Every LNKLST SET automatically contains the SYS1.LINKLIB, SYS1.MIGLIB, SYS1.CSSLIB on top of the concatenation.

2 This statement adds the PDS: ANF.SANFLOAD.

3 This statement activates the LNKLST SET. Only one LNKLST SET can be activated at a time.

The LINK LIST can contain PDSs or PDSEs. PDSs might be catalogued in the master catalog or in a user catalog, while PDSEs must be catalogued in the master catalog. You must specify the volume serial number of a PDS catalogued in a user catalog. PDSEs can have secondary allocation and counts as one extent in the 255 EXTENTs limit (DFSMS/MVS 1.3 or later).

Do not modify the PDS list order because you may find duplicate modules in the LINKST SET. At the end of this article is a short utility to list duplicate members.

DYNAMICALY ADDING A PDS

To add a PDS, enter these MVS commands:

```
SETPROG LNKLST,DEFINE,NAME=mod1,COPYFROM=CURRENT      (1)
SETPROG LNKLST,ADD,NAME=mod1,DSNAME=dsn               (2)
SETPROG LNKLST,ACTIVATE,NAME=mod1                     (3)
```

1 This statement defines a new LNKLST SET named mod1, by copying the active LNKLST SET.

© 1999. Reproduction prohibited. Please inform Xephon of any infringement.

2 This statement adds a new PDS at the end of the LINK LIST. You can place it at the top by adding the ATTOP keyword. You can place it after a particular PDS with the AFTER=dsn keyword.

3 This statement activates the LNKLST SET named mod1. Active tasks continue to use the old LNKLST SET, unless you issue the SETPROG LNKLST,UPDATE command. New tasks will use mod1.

Do not forget to update the PROGxx member for the next IPL!

DYNAMICALLY REMOVING A PDS

To remove a PDS, enter the following MVS commands:

```
SETPROG LNKLST,DEFINE,NAME=mod2,COPYFROM=CURRENT
SETPROG LNKLST,DELETE,NAME=mod2,DSNAME=dsn
SETPROG LNKLST,ACTIVATE,NAME=mod2
```

VIEWING THE ACTIVE LNKLST SET

The D PROG,LNKLST command displays the contents of the last LNKLST SET that has been activated. To know which tasks are using the LNKLST SET mod1, enter the command:

```
D PROG,LNKLST,USERS,NAME=mod1
```

To know which LNKLST SET is being used by jobs whose name starts with PPAI, issue:

```
D PROG,LNKLST,JOBNAME=ppai*
```

To remove a LNKLST SET that is no longer used by active tasks, issue:

```
SETPROG LNKLST,UNDEFINE,NAME=mod2
```

To locate a module in the active LNKLST SET, issue:

```
SETPROG LNKLST,TEST,name=current,modname=module1
```

REPLACEMENT OF THE ENTIRE LINKLIST

The entire LINKLIST can be replaced using a new PROG member:

- Prepare a new member PROGyy in SYS1.PARMLIB, with a different LNKLST SET name, with the old definitions and the

© 1999. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (940) 455 7050, fax (940) 455 2492.

new ones. This member can also contain the APF statements but does not necessarily do so. Do not forget to include an ACTIVATE command.

- Enter 'SET PROG=yy'. Activation can require a few minutes. Active tasks remain with the old LNKLST SET.

LOOKING FOR DUPLICATES IN THE LINK LIST

Here is a SAS program that locates duplicates members in the LINKLIST, by scanning the directories of the PDSs specified in a PROGxx member. This is a sample of the report produced:

```
**** DUPLICATE MODULES IN LINKLIST ****
           (CAN BE ALIASES)
  MODULE : $$$COPYR IN   ISP.SISPSASC
                    IN   SYSP.XPE.LINKLIB
  MODULE : BLDQS    IN   SYS1.VSCLLIB
                    IN   CEE.SCEERUN
  MODULE : CAISERV  IN   SYSP.CA90S.LOADLIB2
                    IN   SYSP.TLMS.LINKLIB
  MODULE : DSNHDECP IN   SYSP.DB2.DSNEXIT
                    IN   SYSP.DB2.DSNLOAD
  MODULE : DSNHLI   IN   SYSP.IMS.RESLIB
                    IN   SYSP.DB2.DSNLOAD
  MODULE : DSN3‡ATH IN   SYSP.DB2.DSNEXIT
                    IN   SYSP.DB2.DSNLOAD
  MODULE : DSN3‡SGN IN   SYSP.DB2.DSNEXIT
                    IN   SYSP.DB2.DSNLOAD
  MODULE : IBMBBCGA IN   SYSP.PLI.PLILINK
                    IN   CEE.SCEERUN
```

The program can be run with the following JCL:

```
  //C100SAS EXEC SAS
  //SASLOG   DD  SYSOUT=Z                              SAS LOG
  //REPORT   DD  SYSOUT=X                              DUPLICATES REPORT
  //PROGXX   DD  DSN=SYS1.PARMLIB(PROG03),DISP=SHR     PROGXX MEMBER
  //TEMP     DD  DSN=xxxx,                             TEMPORARY FILE
  //             DISP=(MOD,DELETE,DELETE),             ==> KEEP DSN
  //             SPACE=(CYL,(5,2))
  //SYSIN    DD  *                                     SAS PROGRAM
```

The program source is shown below:

```
*-------------------------------------------------------------------------+
! PGM SASLNKLT : SCANNING DUPLICATES MODULES IN LINKLIST CONCATENATION !
! THIS PROGRAM READS THE PROGXX MEMBER, THEN READS THE PDS DIRECTORIES !
! TO PRINT THE DUPLICATES MEMBERS.                                      !
*-------------------------------------------------------------------------;
OPTIONS ERRORABEND ;
```

© 1999. Reproduction prohibited. Please inform Xephon of any infringement.

```
* ------------------------------------------------------------------------------;
%MACRO LDIR(B) ;
*                                                                               ;
* SAS MACRO TO LIST PDS DIRECTORY                                               ;
*                                                                               ;
 PROC SOURCE INDD=&B OUTDD=TEMP NODATA NOPRINT ;
 BEFORE &B 45 ALIAS ;
 BEFORE &B 45 ;
* ;
%MEND LDIR ;
* ------------------------------------------------------------------------------;
DATA PDSLIST ;
     INFILE PROGXX ;                          /* TO READ PROGXX MEMBER */
*    DSN EXTRACTION IN PROGXX SYNTAX ;
     INPUT ILINE $ 1-72 ;
     I = INDEX(ILINE,'LNKLST') ;
     IF I NE Ø & I < 1Ø &
        INDEX(ILINE,' ADD ') NE Ø ;
     D = SCAN (ILINE,3,'(') ;
     DSN2 = "'" !! SCAN (D,1,')') !! "'" ;
     CALL EXECUTE('%LDIR('!!DSN2!!')');                /* MACRO CALL */
RUN  ;
* ------------------------------------------------------------------------------;
DATA PDSLIST2 ;                 /* CONVERTING MEMBER LIST TO SAS FILE */
     INFILE TEMP ;
     INPUT PDS $ 1-44 MEMBER $ 45-52 ;
RUN  ;
   * ------------------------------------------------------------------------------;
PROC SORT DATA= PDSLIST2 OUT = PDSLIST3 ; BY MEMBER                             ;
* ------------------------------------------------------------------------------;
DATA _NULL_  ;
*    PRINTING DUPLICATES ;
     SET PDSLIST3  ;
     RETAIN MEMT PDST ; LENGTH PDST $ 44. ;
     FILE REPORT NOTITLE ;
     IF _N_ = 1 THEN
        DO  ;
         MEMT = '        ' ; PDST = '        ' ;
         PUT  à2Ø '**** DUPLICATE MODULES IN LINKLIST ****'//
              à3Ø '(CAN BE ALIASES)'/// ;
        END ;
     IF MEMT = MEMBER THEN
        DO  ;
         PUT à2 ' ' ;
         PUT à2 ' MODULE : ' à12 MEMBER à21 'IN   ' PDST ;
         PUT                            à21 'IN   ' PDS  ;
        END ;
     PDST = PDS ; MEMT = MEMBER ;
RUN ;
* ------------------------------------------------------------------------------;
```


© Xephon 1999

© 1999. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (940) 455 7050, fax (940) 455 2492.

Closing an ‘orphaned’ DCB

THE PROBLEM

In an ISPF/PDF environment, user programs are often LINKed-to rather than being ATTACHed, for example via the ISPEXEC SELECT PGM(*pgmname*) mechanism. Hence such programs run under an ISPF TCB, rather than one of their own. Whilst this avoids the overhead of creating and terminating a task for what is often a trivial transaction, it means that when the user program returns, normal MVS task termination processing does not take place. In particular, if the user program fails, for whatever reason, to close any datasets that it may have OPENed, the open DCB(s) get left behind. Then, when the user tries to run the program again, problems may arise when trying to access the already-open datasets.

The normal way out of this situation is to leave ISPF, so forcing the ISPF task that owns the DCB(s) to terminate, and hence close all its open DCBs. This may, however, not be convenient, and may fail to resolve the problem if the DCB(s) were in dynamically-acquired storage – left to itself task termination deletes such storage before closing open DCBs.

THE SOLUTION

To provide a more elegant method of resolving this problem, I have written a simple program, DCBCLOSE. This program should be invoked by an ISPEXEC SELECT on the same logical screen as the original program that left the open DCB(s), so that it runs under the TCB that owns the DCB(s) – only the task that OPENed a DCB can close it. This is easily accomplished by a simple CLIST of the form:

```
PROC 1 DDNAME
   ISPEXEC SELECT PGM(DCBCLOSE) PARM(&DDNAME)
END
```

where ‘DDNAME’ is the DDNAME of the orphaned DCB. The DCBCLOSE module must be located in a member of the ISPLLIB concatenation for it to be invoked in this way.

© 1999. Reproduction prohibited. Please inform Xephon of any infringement.

DCBCLOSE locates the TIOT and the DEB chain for the active TCB, and then scans the DEB chain for a DEB whose TIOT entry has the specified ‘DDNAME’. If found, a CLOSE macro is issued for the associated DCB and a message of the following form is issued:

```
DCBCLOSE - Dataset opened to DDNAME has been successfully closed
```

After this DCBCLOSE exits.

OPERATIONAL ENVIRONMENT

DCBCLOSE has no special authorization requirements, and may be link-edited into any suitable load library that is part of the ISPLLIB concatenation.

DCBCLOSE was written for use on an MVS/XA 2.2.3 system with ISPF/PDF Version 2, and has since been used on an MVS/ESA 4.2.2 system with ISPF/PDF Version 3, and an MVS/ESA 5.1.0 system with ISPF/PDF Version 4.

DCBCLOSE

```
         TITLE 'DCBCLOSE Close an Orphaned DCB'
*********************************************************************
*  PROGRAM DCBCLOSE
*
*  This simple program searches for and closes an 'orphan' DCB, given
*  the DDNAME that was used to open it. It is not an unusual thing in
*  an ISPF environment,  where programs tend  to  be LINKed-to rather
*  than ATTACHed, that an abending program terminates without closing
*  its datasets.  This program provides an alternative way of closing
*  such datasets to leaving  and  re-entering the ISPF environment.
*
*  In order to  ensure  that  DCBCLOSE executes under the same TCB as
*  the program that opened the orphan DCB,   it should be invoked via
*  the ISPF ISPEXEC SELECT service  in  the same logical screen.  For
*  example the following CLIST would suffice :
*
*           PROC 1 DDNAME
*           ISPEXEC SELECT PGM(DCBCLOSE) PARM(&DDNAME)
*           END
*
*  Where  &DDNAME  is  the  1-8 character DDNAME of the dataset to be
*  closed.   The  DCBCLOSE load module must be in a load library that
*  is part of the ISPLLIB concatenation, but otherwise has no special
*  attributes.
*
```


© 1999. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (940) 455 7050, fax (940) 455 2492.

```
*  Environmental requirements:
*
*    STATE    : Problem
*    KEY      : 8
*    APF      : No
*    AMODE    : 31
*    RMODE    : ANY
*    LOCATION : Private library in ISPLLIB concatenation
************************************************************************
         EJECT
DCBCLOSE CSECT
DCBCLOSE AMODE 31
DCBCLOSE RMODE ANY
R0       EQU   0                        *
R1       EQU   1                        * PARM FIELD ADDRESS
R2       EQU   2                        * DDNAME LENGTH
R3       EQU   3                        *
R4       EQU   4                        *
R5       EQU   5                        *
R6       EQU   6                        *
R7       EQU   7                        *
R8       EQU   8                        * @(DCB)
R9       EQU   9                        * @(DEB)
R10      EQU   10                       * @(TIOT)
R11      EQU   11                       * @(TCB)
R12      EQU   12                       * BASE REGISTER
R13      EQU   13                       * SAVEAREA
R14      EQU   14                       * RETURN ADDRESS
R15      EQU   15                       * ENTRY ADDRESS/RETURN CODE
B     START-*(R15)                      * BRANCH TO CODE
         DC    AL1(NT2-NT1)             * LENGTH OF NAME TEXT
NT1      EQU   *
         DC    C'DCBCLOSE'              * MODULE NAME
NT2      EQU   *
         DC    C' '
         DC    CL8'&SYSDATE'            * DATE
         DC    C' '
         DC    CL5'&SYSTIME'            * TIME
         DS    0F                       * ALIGN TO FULL WORD BOUNDARY
************************************************************************
*        ADDRESSABILITY AND LINKAGE
************************************************************************
START    EQU   *
         STM   R14,R12,12(R13)          * SAVE REGISTERS IN HSA
         LR    R12,R15                  * LOAD BASE REGISTER R12
         USING DCBCLOSE,R12             * DEFINE R12 AS BASE REGISTER
         LR    R11,R13                  * R11 = ADDRESS OF HSA
         LA    R13,SAVEAREA             * R13 = ADDRESS OF LSA
         ST    R11,4(R13)               * STORE HSA ADDRESS
         ST    R13,8(R11)               * STORE LSA ADDRESS
* GET DDNAME FROM PARM FIELD
         L     R1,0(R1)                 * R1 = ADDRESS OF PARM FIELD
```

© 1999. Reproduction prohibited. Please inform Xephon of any infringement.

```
         LH    R2,0(R1)                  * R2 = LENGTH OF PARM FIELD
         LTR   R2,R2                     * TEST VALUE
         BZ    NOPARM                    * ERROR IF NO PARM SPECIFIED
         CH    R2,H8                     * MAX LENGTH IS 8 CHARACTERS
         BH    BADPARM                   * IF LONGER ASSUME ERROR
         MVI   DDNAME,C' '               * BLANK OUT ...
         MVC   DDNAME+1(7),DDNAME        * ... DDNAME FIELD
         BCTR  R2,0                      * MOVE DDNAME ...
         EX    R2,MOVEDDN                * ... FROM PARM FIELD
         EJECT
*********************************************************************
*        SEARCH FOR THE DCB AND CLOSE IT
*********************************************************************
* GET THE TIOT AND DEB CHAIN ADDRESSES FROM THE CURRENT TCB
         USING PSA,R0
         L     R11,PSATOLD               * TCB ADDRESS ...
         DROP  R0
         USING TCB,R11                   * ... AND ADDRESSABILITY
         ICM   R10,B'1111',TCBTIO        * TIOT ADDRESS
         BZ    NODCB                     * NO TIOT MEANS NO DCB
         ICM   R9,B'1111',TCBDEB         * DEB ADDRESS
         BZ    NODCB                     * NO DEB MEANS NO DCB
         DROP  R11                       * FINISHED WITH TCB
         USING DEBBASIC,R9               * DEB ADDRESSABILITY
* SEARCH THE DEB CHAIN FOR A DEB WHOSE TIOT ENTRY HAS THE RIGHT DDNAME
         SR    R8,R8                     * CLEAR R8
DEBLOOP  EQU   *
         ICM   R8,B'0111',DEBDCBB        * DCB ADDRESS ...
         USING IHADCB,R8                 *     ... AND ADDRESSABILITY
         LH    R7,DCBTIOT                * OFFSET TO DD ENTRY IN TIOT
         DROP  R8                        * FINISHED WITH DCB
         AR    R7,R10                    * TIOT ENTRY ADDRESS ...
         USING TIOENTRY,R7               * ... AND ADDRESSABILITY
         CLC   TIOEDDNM,DDNAME           * DDNAMES MATCH?
         BE    CLOSE                     * YES, SO GO AND CLOSE DCB
         DROP  R7                        * FINISHED WITH TIOT ENTRY
         ICM   R9,B'0111',DEBDEBB        * @(NEXT DEB)
         BNZ   DEBLOOP                   * LOOP BACK UNTIL END
         B     NODCB                     * DCB NOT FOUND
         DROP  R9                        * FINISHED WITH DEB
* WE HAVE FOUND THE DCB OPENED TO THE SPECIFIED DDNAME, SO CLOSE IT
CLOSE    EQU   *
         CLOSE ((R8)),MODE=31            * CLOSE DCB
* ... AND TELL THE CALLER ALL IS WELL
         MVC   WTO1+44(8),DDNAME
WTO1     WTO   'DCBCLOSE - Dataset opened to DDNAME ........ has been s+
               uccessfully closed',ROUTCDE=11
         SR    R15,R15                   * ZERO RETURN CODE
         EJECT
*********************************************************************
*        ALL DONE, SO RETURN TO CALLER
*********************************************************************
```


© 1999. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (940) 455 7050, fax (940) 455 2492.

```
RETURN   EQU   *
         L     R13,4(R13)               * RESTORE HSA ADDRESS
         L     R14,12(R13)              * RESTORE R14
         LM    R0,R12,20(R13)           * RESTORE R0-R12
         BR    R14                      * AND RETURN
         EJECT
*********************************************************************
*        ERROR CONDITIONS
*********************************************************************
NOPARM   EQU   *
         WTO   'DCBCLOSE - Please supply the DDNAME of the dataset you +
               wish to close',ROUTCDE=11
         LA    R15,4                    * NO PARM  - SET RC=04
         B     RETURN                   * RETURN
BADPARM  EQU   *
         WTO   'DCBCLOSE - Supplied DDNAME is more than 8 characters lo+
               ng',ROUTCDE=11
         LA    R15,8                    * BAD PARM - SET RC=08
         B     RETURN                   * RETURN
NODCB    EQU   *
         MVC   WTO4+51(8),DDNAME
WTO4     WTO   'DCBCLOSE - No dataset found open to DDNAME ........',  +
               ROUTCDE=11
         LA    R15,12                   * NO DCB   - SET RC=12
         B     RETURN                   * RETURN
         EJECT
*********************************************************************
*        CONSTANTS AND DATA AREAS
*********************************************************************
         DS    0D
         DC    CL8'SAVEAREA'
SAVEAREA DC    18F'0'                   * SAVE AREA
DDNAME   DS    CL8                      * DDNAME
H8       DC    H'8'                     * MAXIMUM DDNAME LENGTH
MOVEDDN  MVC   DDNAME(0),2(R1)          * EXECUTED MVC FOR DDNAME
* SYSTEM CONTROL BLOCK DSECTS
         PRINT NOGEN
         IHAPSA LIST=NO
         IKJTCB LIST=NO
         DSECT
         IEFTIOT1                       * TIOT MAPPING MACRO
         IEZDEB                         * DEB MAPPING MACRO
         DCBD  DSORG=PS,DEVD=DA         * DCB MAPPING MACRO
         END
```

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

© Xephon 1999

© 1999. Reproduction prohibited. Please inform Xephon of any infringement.

Generic tape read and write routines

INTRODUCTION

From time to time, I have been presented with a tape to read without being given any information as to its format. Trying to work out how to read it by trial and error can be a frustrating and time-consuming business. To simplify this task, I have written a generic tape reading routine, TPREAD, which will read a tape written in any format. This allows the contents of the tape to be dumped and visually inspected, so allowing a decision to be made as to how it should be used. I have also written a companion routine, TPWRITE, which, when used in conjunction with TPREAD, can be used to make an image copy of any tape. An example of how this can be done is described below.

TPREAD

The TPREAD routine builds a channel program to read a single block at a time and executes it via the EXCP macro. If the tape is mounted for bypass label processing (assuming that your security policy permits this), TPREAD will read the header and trailer labels, if any, as normal data. The HDR2 label of a standard labelled tape in particular contains useful information about the file, such as the record format, block size, and record length. Tape marks are noted as such and skipped – TPREAD will read until the end of the recorded information on the tape. TPREAD can even be used to read DFSMSdss dump tapes, which cannot be read by normal access methods because they are written with 64KB blocks; normal access methods, such as QSAM, are limited to a maximum blocksize of 32KB.

TPWRITE

The TPWRITE routine builds a channel program to write a single block of data or a tape mark, and executes it via the EXCP macro. If being used in conjunction with TPREAD to copy a tape, the input tape is being read with bypass label processing, the output tape was mounted as non-labelled, and every record read by TPREAD is written unchanged by TPWRITE, the output tape will be an exact image of the input tape, including all the header and trailer labels.

© 1999. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (940) 455 7050, fax (940) 455 2492.

A note of caution – when MVS demounts the output tape, it will sense the (new) volume (VOL1) label, and, if the tape is under any form of automated tape management (eg in an automated tape library), conflicts can occur as the tape management software will suddenly have two tapes with the same volume serial number. To avoid such problems, the output tape at least is best mounted on a manual drive. Another possibility would be for the tape copy program to alter the volser in the VOL1 label rather than writing a duplicate of the input.

SUGGESTED USAGE

TPREAD and TPWRITE can be called from any programming language that supports standard OS/370 linkage conventions, and have no addressing or residency mode restrictions nor special authorization requirements. An outline of a typical program to read and dump a tape for inspection is as follows:

1 Allocate the tape, for bypass label processing if required and permitted.

2 Call TPREAD to read the next block. Tape marks and the physical end of tape are presented with special return codes, otherwise a return code of zero means a data block has been successfully read.

3 Display the block in an appropriate manner – for example use QSAM to write it to a RECFM=U, BLKSIZE=32760 dataset for later BROWSEing, or to JES SYSOUT for inspection via SDSF or IOF.

4 Loop back to step 3 until the end-of-tape return code is received.

An outline of a typical program to make an image copy of a tape would be as follows:

1 Allocate the input tape, for bypass label processing if required and permitted.

2 Allocate the (non-labelled) output tape.

3 Call TPREAD to read the next block. Tape marks and the physical end of tape are presented with special return codes, otherwise a return code of zero means a data block has been successfully read.

© 1999. Reproduction prohibited. Please inform Xephon of any infringement.

4 Call TPWRITE to write the label record, data block, or tape mark, as appropriate. If required, change the volser in the VOL1 label before writing it.

5 Loop back to step 3 until the end-of-tape return code is received.

6 Call TPWRITE to close and demount the output tape.

OPERATIONAL ENVIRONMENT

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

The versions of TPREAD and TPWRITE presented here should run under any version of MVS/XA or MVS/ESA, and DFP or DFSMS/MVS; they are known to work on MVS/XA 2.2.3 + DFP 2.4, MVS/ESA 4.2.2 + DFP 3.3, and MVS/ESA 5.1.0 + DFSMS/MVS 1.2.0. They have been used for open reel tapes (3420) and for cartridge tapes (3480, 3490E).

TPREAD

```
         TITLE 'TPREAD - Generic Tape Read Routine'
*********************************************************************
*  Subroutine TPREAD
*  This routine is designed  to  read  a tape written in an arbitrary
*  format.   It is not sensitive to EOF markers,  and, when used with
*  bypass label processing, will read the entire contents of the tape
*  including header and trailer labels.
*
*  This routine is designed  for  robustness rather than efficiency -
*  it is intended more as a diagnostic or recovery tool.  For example
*  it can be used to copy  a  damaged or otherwise unreadable dataset
*  to a BROWSEable copy.   In this context,  the copy dataset is best
*  allocated with RECFM=U and BLKSIZE=32760.  Once this readable copy
*  has been made, recovery actions specific to the nature of the data
*  can be taken, using normal access methods.
*
*  This routine may  be called from any high-level language that uses
*  standard OS/370 linkage conventions.  It has no addressing or res-
```


© 1999. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (940) 455 7050, fax (940) 455 2492.

```
*  idency mode restrictions.
*
*  ARGUMENTS :  CALL TPREAD(DDNAME,NBYTES,RECORD,IOSTAT)
*  ————   ————————————————————
*
*    DDNAME  :  DDNAME of pre-allocated dataset              (  INPUT )
*    NBYTES  :  Number of bytes read (if IERR=Ø)             ( OUTPUT )
*    RECORD  :  Record                                       ( OUTPUT )
*    IOSTAT  :  Error flag                                   ( OUTPUT )
*
*  The value of IOSTAT should always be checked on return. the values
*  IOSTAT may have are:
*
*     IOSTAT =  Ø : Record read successfully; length is in NBYTES.
*               4 : Dataset open failed (probably not allocated).
*               8 : EOF (zero length record) - move to next record.
*              2Ø : Unit check - end of tape
*
*  Note that NBYTES is only set if IOSTAT is zero,  and should not be
*  used unless that is the case.   Providing that it is set,   RECORD
*  will contain NBYTES bytes of data.
*
*  Operational requirements :
*
*    STATE    : Problem
*    KEY      : 8
*    APF      : No
*    AMODE    : 31
*    RMODE    : No restriction
*    LOCATION : Callable subroutine
*********************************************************************
         EJECT
TPREAD   CSECT
TPREAD   AMODE 31
TPREAD   RMODE ANY
RØ       EQU   Ø                        * WORK REGISTER
R1       EQU   1                        * @(ARGUMENT LIST)
R2       EQU   2                        * @(DDNAME)
R3       EQU   3                        * @(NBYTES)
R4       EQU   4                        * @(BUFFER)
R5       EQU   5                        * @(IOSTAT)
R6       EQU   6                        *
R7       EQU   7                        *
R8       EQU   8                        * WORK REGISTER
R9       EQU   9                        * WORK REGISTER
R1Ø      EQU   1Ø                       * WORK REGISTER
R11      EQU   11                       * WORK REGISTER
R12      EQU   12                       * BASE REGISTER
R13      EQU   13                       * OUR SAVEAREA
R14      EQU   14                       * RETURN ADDRESS
```

© 1999. Reproduction prohibited. Please inform Xephon of any infringement.

```
R15      EQU   15                        * ENTRY ADDRESS/RETURN CODE
         USING *,R15                     * ADDRESSABILITY
         B     START                     * BRANCH TO START OF CODE
         DC    AL1(LASTL-FIRSTL)         * LENGTH OF HEADER TEXT
FIRSTL   EQU   *
         DC    CL8'TPREAD  '
LASTL    EQU   *
         DC    C' '
         DC    CL8'&SYSDATE'
         DC    C' '
         DC    CL5'&SYSTIME'
         DROP  R15                       * FINISHED WITH R15
         DS    ØF                        * ALIGN TO FULL WORD BOUNDARY
*********************************************************************
* ADDRESSABILITY AND LINKAGE
*********************************************************************
START    EQU   *
         STM   R14,R12,12(R13)           * SAVE REGISTERS IN HSA
         LR    R12,R15                   * LOAD BASE REGISTER
         USING TPREAD,R12                * AND DEFINE ADDRESSIBILITY
         LR    R11,R13                   * R11 = ADDRESS OF HSA
         LA    R13,SAVEAREA              * R13 = ADDRESS OF LSA
         ST    R11,4(R13)                * STORE HSA ADDRESS
         ST    R13,8(R11)                * STORE LSA ADDRESS
* GET ARGUMENT ADDRESSES
         LM    R2,R5,Ø(R1)               * LOAD ARG ADDRESSES
* IS THIS THE FIRST CALL ?
         ICM   R1,B'1111',ADEB           * FIRST CALL?
         BNZ   DOREAD                    * NO
         EJECT
*********************************************************************
*        ON THE FIRST CALL, PERFORM SOME ONCE-OFF INITIALIZATION
*        PROCESSING
*********************************************************************
* IF NEED BE GET BELOW-LINE STORAGE FOR DCB, IOB, AND CHANNEL PROGRAM
         ICM   R1,B'1111',ATPDCB         * IF A DCB ...
         BZ    GETDCB                    * ... ALREADY ...
         ST    R1,ADCB                   *     ... OPEN ...
         B     GETIOB                    *         ... USE IT
GETDCB   EQU   *
         TM    ADCB,X'FF'                * ARE WE ABOVE THE LINE?
         BZ    BUILDCBS                  * IF NOT NO NEED TO MOVE DCB
         GETMAIN RU,LV=LDCB,BNDRY=DBLWD,LOC=(BELOW,ANY)
         ST    R1,ADCB                   * SAVE BELOW-LINE DCB ADDRESS
         MVC   Ø(LDCB,R1),DCB            * MOVE DCB BELOW LINE
GETIOB   EQU   *
         TM    AIOB,X'FF'                * ARE WE ABOVE THE LINE?
         BZ    BUILDCBS                  * IF NOT NO NEED TO MOVE IOB
         GETMAIN RU,LV=LIOB,BNDRY=DBLWD,LOC=(BELOW,ANY)
         ST    R1,AIOB                   * SAVE IOB ADDRESS
```


© 1999. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (940) 455 7050, fax (940) 455 2492.

```
         MVC   Ø(LIOB,R1),IOBAREA        * MOVE IOB BELOW LINE
         GETMAIN RU,LV=LCHPROG,BNDRY=DBLWD,LOC=(BELOW,ANY)
         ST    R1,ACHPROG                * SAVE @(CHANNEL PROGRAM)
         MVC   Ø(LCHPROG,R1),CHPROG      * MOVE CHPROG BELOW LINE
*
*-------------------------------------------------------------------
*        COMPLETE DCB, IOB, AND CHANNEL PROGRAM
*-------------------------------------------------------------------
*
BUILDCBS EQU   *
         L     R11,ADCB                  * R11 = DCB ADDRESS
         USING IHADCB,R11                * DEFINE DCB ADDRESSABILITY
         ICM   R1,B'1111',ATPDCB         * IF DCB ALREADY OPEN ...
         BNZ   BUILDIOB                  * ... DON'T INTERFERE WITH IT
         MVC   DCBDDNAM,Ø(R2)            * MOVE DDNAME INTO DCB
BUILDIOB EQU   *
         L     R1Ø,AIOB                  * R1Ø = IOB ADDRESS
         USING IOB,R1Ø                   * IOB ADDRESSABILITY
         MVI   IOBFLAG1,X'42'            * CMND CHAINING,UNRELATED
         STCM  R11,B'Ø111',IOBDCBPT      * INSERT @(DCB) INTO IOB
         L     R9,ACHPROG                * @(CHANNEL PROGRAM)
         STCM  R9,B'Ø111',IOBSTART       * INSERT @(CH PROG) INTO IOB
         LA    R8,IOBECB                 * @(IOBECB)
         STCM  R8,B'Ø111',IOBECBPT       * INSERT @(IOBECB) INTO IOB
         LA    R8,8(R9)                  * INSERT @(IDAW) ...
         STCM  R8,B'Ø111',1(R9)          * ... INTO RD CCW
         DROP  R1Ø                       * FINISHED WITH IOB
*
*-------------------------------------------------------------------
*        OPEN THE DATASET
*-------------------------------------------------------------------
*
         ICM   R1,B'1111',ATPDCB         * IF DCB ALREADY OPEN ...
         BNZ   OPENED                    * ... DON'T OPEN IT AGAIN
         OPEN  ((R11),INPUT),MODE=31     * OPEN DATASET FOR INPUT
         TM    DCBOFLGS,DCBBIT3          * BIT 3 SHOULD BE 1
         BZ    ERROR4                    * ITS NOT SO AN ERROR OCCURRED
* GET DEB ADDRESS FROM DCB AND SAVE IT
OPENED   EQU   *
         SR    R1,R1                     * R1 = ...
         ICM   R1,B'Ø111',DCBDEBA        * ... DEB ADDRESS
         ST    R1,ADEB                   * SAVE IT
         ICM   R1,B'1111',ATPDCB         * IF DCB ALREADY OPEN ...
         BNZ   DOREAD                    * ... DON'T SAVE @(DCB)
         ST    R11,ATPDCB                * SAVE @(DCB) FOR TPOINT
         DROP  R11                       * FINISHED WITH DCB
         EJECT
*********************************************************************
*        READ DATA
*********************************************************************
DOREAD   EQU   *
```

© 1999. Reproduction prohibited. Please inform Xephon of any infringement.

```
         L     R9,ACHPROG                * INSERT BUFFER ADDRESS ...
         ST    R4,8(R9)                  * ... INTO IDAW
         L     R1Ø,AIOB                  * R1Ø = IOB ADDRESS
         USING IOB,R1Ø                   * IOB ADDRESSABILITY
         XC    IOBECB,IOBECB             * CLEAR ECB
         EXCP  (R1Ø)                     * EXECUTE CHANNEL PROGRAM
         WAIT  ECB=IOBECB                * WAIT ON IOBECB
         TM    IOBECB,X'7F'              * TEST FOR SUCCESSFULL READ
         BO    READOK                    * MATCH MEANS READ OK
*
*-------------------------------------------------------------------
*        I/O ERROR DETECTED. ANALYSE THE SITUATION:
*-------------------------------------------------------------------
*
* UNIT CHECK PROBABLY MEANS WE HAVE HIT BLANK TAPE
         TM    IOBCSW+3,X'Ø2'            * UNIT CHECK?
         BNO   TRYEOF                    * NO, SO TRY FOR REAL EOF
         B     ERROR2Ø                   * ANYTHING ELSE IS FATAL ERROR
* A RECORD WITH ZERO DATA LENGTH IS AN END-OF-FILE (TAPE MARK)
TRYEOF   EQU   *
         SR    R9,R9                     * R9 = ...
         ICM   R9,B'ØØ11',R+6            * ... CCW COUNT
         SR    R8,R8                     * R8 = ...
         ICM   R8,B'ØØ11',IOBCSW+5       * ... RESIDUAL COUNT
         CR    R9,R8                     * COMPARE CCW & RESIDUAL COUNT
         BE    EOF                       * EQUAL MEANS EOF (TAPE MARK)
         B     ERROR2Ø                   * ANYTHING ELSE IS FATAL ERROR
*
*-------------------------------------------------------------------
*        RECORD READ OK. COMPUTE # BYTES READ
*-------------------------------------------------------------------
*
READOK   EQU   *
         SR    R9,R9                     * R9 = ...
         ICM   R9,B'ØØ11',R+6            * ... CCW COUNT
         SR    R8,R8                     * R8 = ...
         ICM   R8,B'ØØ11',IOBCSW+5       * ... RESIDUAL COUNT
         SR    R9,R8                     * # BYTES READ
         ST    R9,Ø(R3)                  * SAVED FOR CALLER
         SR    R15,R15                   * RC = Ø
         DROP  R1Ø                       * FINISHED WITH IOB
         EJECT
*********************************************************************
*        RETURN TO CALLER
*********************************************************************
RETURN   EQU   *
         ST    R15,Ø(R5)                 * STORE IOSTAT FOR CALLER
         L     R13,4(R13)                * RESTORE ADDRESS OF HSA
         L     R14,12(R13)               * RESTORE R14
         LM    RØ,R12,2Ø(R13)            * RESTORE RØ-R12
         BR    R14                       * AND RETURN
         EJECT
```


© 1999. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (940) 455 7050, fax (940) 455 2492.

```
*************************************************************************
* ERROR CONDITIONS
*************************************************************************
ERROR4   EQU   *
         LA    R15,4                    * OPEN FAILURE
         B     RETURN                   * RETURN
EOF      EQU   *
         LA    R15,8                    * EOF (TAPE MARK)
         B     RETURN                   * RETURN
ERROR2Ø  EQU   *
         LA    R15,2Ø                   * I/O ERROR
         B     RETURN                   * RETURN
         EJECT
*************************************************************************
*        CONSTANTS, VARIABLES AND DATA AREAS
*************************************************************************
         DS    ØD
         DC    CL8'SAVEAREA'
SAVEAREA DS    18F
         DS    ØF
         ENTRY TPDCB
TPDCB    EQU   *
ATPDCB   DC    A(Ø)
*
ADCB     DC    A(DCB)
ADEB     DC    A(Ø)
AIOB     DC    A(IOBAREA)
ACHPROG  DC    A(CHPROG)
         DS    ØF
DCB      DCB   DDNAME=DUMMY,DSORG=PS,DEVD=TA,MACRF=E
LDCB     EQU   *-DCB
         DS    ØF
IOBAREA  DC    (LIOB)X'ØØ'
         DS    ØD
CHPROG   EQU   *
R        CCW   X'Ø2',IDAW,X'24',65535    * READ
IDAW     DS    F                        * INDIRECT DATA ADDRESS WORD
LCHPROG  EQU   *-CHPROG
IOB      DSECT
IOBFLAG1 DS    XL1                      * CHAINING/UNRELATED BITS
IOBFLAG2 DS    XL1                      * NOT USED HERE
IOBSENSE DS    ØXL2                     * SENSE BYTES
IOBSENSØ DS    XL1                      *   SENSE BYTE 1
IOBSENS1 DS    XL1                      *   SENSE BYTE 2
IOBECBCC DS    XL1                      * FIRST BYTE OF COMP. CODE
IOBECBPT DS    AL3                      * ECB ADDRESS
IOBFLAG3 DS    XL1                      * SYSTEM USE ONLY
IOBCSW   DS    XL7                      * CHANNEL STATUS WORD
IOBSIOCC DS    XL1                      * START SUBCHANNEL DATA
IOBSTART DS    AL3                      * CHANNEL PROGRAM ADDRESS
         DS    XL1                      * RESERVED
IOBDCBPT DS    AL3                      * DCB ADDRESS
```

© 1999. Reproduction prohibited. Please inform Xephon of any infringement.

```
IOBRESTR DS    XL1                      * USED FOR ERROR RECOVERY
         DS    XL3                      * USED FOR ERROR RECOVERY
IOBINCAM DS    XL2                      * USED FOR MAG TAPE ONLY
         DS    XL2                      * RESERVED
IOBECB   DS    F                        * ECB
LIOB     EQU   *-IOB
         PRINT NOGEN
         DCBD  DSORG=PS,DEVD=TA         * DCB MAPPING MACRO
         END
```

TPWRITE

```
         TITLE 'TPWRITE - Generic Tape Write Routine'
*********************************************************************
*  SUBROUTINE TPWRITE
*  This routine is designed  to  write a tape in an arbitrary format.
*  Its intended use is for making image copies of tapes of any  label
*  type,   and containing an arbitrary number of files.  If the input
*  tape is labelled,   is  read using the TPREAD routine under bypass
*  label processing,  and every record returned by TPREAD is  written
*  directly to the output tape by this routine,   the  resulting tape
*  will be an exact copy, including all header and trailer labels.
*
*  This routine may  be called from any high-level language that uses
*  standard OS/370 linkage conventions.  It has no addressing or res-
*  idency mode restrictions.
*
*
*  ARGUMENTS : CALL TPWRITE(DDNAME,NBYTES,RECORD,IOSTAT)
*  ____   ____________________
*
*    DDNAME  :  DDNAME of pre-allocated dataset             (  INPUT )
*    NBYTES  :  Number of bytes to write (see note)         ( IN/OUT )
*    RECORD  :  Record                                      ( OUTPUT )
*    IOSTAT  :  Error flag                                  ( OUTPUT )
*
*  Note: NBYTES should have one of the following values:
*        1) > 0 : Write data record of this length
*        2)   0 : Write a tape mark (EOF)
*        3) < 0 : Close the file
*
*        In the case of a data check (IOSTAT = 16),   NBYTES  will on
*        return contain the number of bytes written to that point.
*
*  Note: RECORD should contain NBYTES bytes  of  data  to be written,
*        except in the  case  of  a  write-tape-mark request or close
*        request, in which case it is ignored.
*
*  The value of IOSTAT should always be checked on return. The values
*  IOSTAT may have are:
*
```


© 1999. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (940) 455 7050, fax (940) 455 2492.

```
*     IOSTAT =  Ø : Record written successfully
*               4 : Dataset open failed (probably not allocated).
*               8 : Unit exception - end of tape
*              12 : Unit check - command reject (write protect)
*              16 : Unit check - data check (NBYTES = bytes written)
*              2Ø : Unit check - any other error condition
*
*  Operational requirements:
*
*    STATE    : Problem
*    KEY      : 8
*    APF      : No
*    AMODE    : 31
*    RMODE    : No restriction
*    LOCATION : Callable subroutine
*********************************************************************
         EJECT
TPWRITE  CSECT
TPWRITE  AMODE 31
TPWRITE  RMODE ANY
RØ       EQU   Ø                        * WORK REGISTER
R1       EQU   1                        * @(ARGUMENT LIST)
R2       EQU   2                        * @(DDNAME)
R3       EQU   3                        * @(NBYTES)
R4       EQU   4                        * @(BUFFER)
R5       EQU   5                        * @(IOSTAT)
R6       EQU   6                        *
R7       EQU   7                        *
R8       EQU   8                        * WORK REGISTER
R9       EQU   9                        * WORK REGISTER
R1Ø      EQU   1Ø                       * WORK REGISTER
R11      EQU   11                       * WORK REGISTER
R12      EQU   12                       * BASE REGISTER
R13      EQU   13                       * OUR SAVEAREA
R14      EQU   14                       * RETURN ADDRESS
R15      EQU   15                       * ENTRY ADDRESS/RETURN CODE
USING *,R15                             * ADDRESSABILITY
         B     START                    * BRANCH TO START OF CODE
         DC    AL1(LASTL-FIRSTL)        * LENGTH OF HEADER TEXT
FIRSTL   EQU   *
         DC    CL8'TPWRITE '
LASTL    EQU   *
         DC    C' '
         DC    CL8'&SYSDATE'
         DC    C' '
         DC    CL5'&SYSTIME'
         DROP  R15                      * FINISHED WITH R15
         DS    ØF                       * ALIGN TO FULL WORD BOUNDARY
*********************************************************************
*        ADDRESSABILITY AND LINKAGE
*********************************************************************
START    EQU   *
         STM   R14,R12,12(R13)          * SAVE REGISTERS IN HSA
```

© 1999. Reproduction prohibited. Please inform Xephon of any infringement.

```
         LR    R12,R15                  * LOAD BASE REGISTER
         USING TPWRITE,R12              * AND DEFINE ADDRESSIBILITY
         LR    R11,R13                  * R11 = ADDRESS OF HSA
         LA    R13,SAVEAREA             * R13 = ADDRESS OF LSA
         ST    R11,4(R13)               * STORE HSA ADDRESS
         ST    R13,8(R11)               * STORE LSA ADDRESS
* GET ARGUMENT ADDRESSES
*
         LM    R2,R5,Ø(R1)              * LOAD ARG ADDRESSES
* IS THIS THE FIRST CALL ?
         ICM   R1,B'1111',ADEB          * FIRST CALL?
         BNZ   DOWRITE                  * NO
         EJECT
*********************************************************************
*        ON THE FIRST CALL, PERFORM SOME ONCE-OFF INITIALISATION
*        PROCESSING
*********************************************************************
* IF NEED BE GET BELOW-LINE STORAGE FOR DCB, IOB, AND CHANNEL PROGRAM
*
         TM    ADCB,X'FF'               * ARE WE ABOVE THE LINE?
         BZ    BUILDCBS                 * IF NOT NO NEED TO MOVE DCB
         GETMAIN RU,LV=LDCB,BNDRY=DBLWD,LOC=(BELOW,ANY)
         ST    R1,ADCB                  * SAVE BELOW-LINE DCB ADDRESS
         MVC   Ø(LDCB,R1),DCB           * MOVE DCB BELOW LINE
         GETMAIN RU,LV=LIOB,BNDRY=DBLWD,LOC=(BELOW,ANY)
         ST    R1,AIOB                  * SAVE IOB ADDRESS
         MVC   Ø(LIOB,R1),IOBAREA       * MOVE IOB BELOW LINE
         GETMAIN RU,LV=LCHPROG,BNDRY=DBLWD,LOC=(BELOW,ANY)
         ST    R1,ACHPROG               * SAVE @(CHANNEL PROGRAM)
         MVC   Ø(LCHPROG,R1),CHPROG     * MOVE CHPROG BELOW LINE
*
*-------------------------------------------------------------------
*        COMPLETE DCB, IOB, AND CHANNEL PROGRAM
*-------------------------------------------------------------------
*
BUILDCBS EQU   *
         L     R11,ADCB                 * R11 = DCB ADDRESS
         USING IHADCB,R11               * DEFINE DCB ADDRESSABILITY
         MVC   DCBDDNAM,Ø(R2)           * MOVE DDNAME INTO DCB
         L     R1Ø,AIOB                 * R1Ø = IOB ADDRESS
         USING IOB,R1Ø                  * IOB ADDRESSABILITY
         MVI   IOBFLAG1,X'42'           * CMND CHAINING,UNRELATED
         STCM  R11,B'Ø111',IOBDCBPT     * INSERT @(DCB) INTO IOB
         L     R9,ACHPROG               * @(CHANNEL PROGRAM)
         STCM  R9,B'Ø111',IOBSTART      * INSERT @(CH PROG) INTO IOB
         LA    R8,IOBECB                * @(IOBECB)
         STCM  R8,B'Ø111',IOBECBPT      * INSERT @(IOBECB) INTO IOB
         LA    R8,16(R9)                * INSERT @(IDAW) ...
         STCM  R8,B'Ø111',W+1           * ... INTO WRITE CCW
         DROP  R1Ø                      * FINISHED WITH IOB
*
*-------------------------------------------------------------------
```


© 1999. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (940) 455 7050, fax (940) 455 2492.

```
*        OPEN THE DATASET
*-------------------------------------------------------------------
*
         OPEN  ((R11),OUTPUT),MODE=31    * OPEN DATASET FOR OUTPUT
         TM    DCBOFLGS,DCBBIT3          * BIT 3 SHOULD BE 1
         BZ    ERROR4                    * ITS NOT SO AN ERROR OCCURRED
* GET DEB ADDRESS FROM DCB AND SAVE IT
         SR    R1,R1                     * R1 = ...
         ICM   R1,B'Ø111',DCBDEBA        * ... DEB ADDRESS
         ST    R1,ADEB                   * SAVE IT
         DROP  R11                       * FINISHED WITH DCB
         EJECT
*************************************************************************
*        WRITE DATA
*************************************************************************
DOWRITE  EQU   *
         ICM   R6,B'1111',Ø(R3)          * NBYTES
         BNP   NODATA                    * IF Ø OR -VE NO DATA
         L     R9,ACHPROG                * @(CHANNEL PROGRAM)
         MVC   Ø(8,R9),W                 * MOVE IN WRITE CCW
         STH   R6,6(R9)                  * UPDATE CCW BYTE COUNT
         ST    R4,16(R9)                 * UPDATE IDAW WITH @(BUFFER)
         B     EXCP                      * CHANNEL PROGRAM IS READY
NODATA   EQU   *
         BM    CLOSE                     * NYBTES -VE MEANS CLOSE
         L     R9,ACHPROG                * @(CHANNEL PROGRAM)
         MVC   Ø(8,R9),WTM               * MOVE IN WRITE-TAPE-MARK CCW
*
* EXECUTE CHANNEL PROGRAM
*
EXCP     EQU   *
         L     RIØ,AIOB                  * RIØ = IOB ADDRESS
         USING IOB,RIØ                   * IOB ADDRESSABILITY
         XC    IOBECB,IOBECB             * CLEAR ECB
         EXCP  (RIØ)                     * EXECUTE CHANNEL PROGRAM
         WAIT  ECB=IOBECB                * WAIT ON IOBECB
         SR    R15,R15                   * CLEAR RETURN CODE
         TM    IOBECB,X'7F'              * TEST FOR SUCCESSFULL WRITE
         BO    RETURN                    * MATCH MEANS WRITE OK
*
*-------------------------------------------------------------------
*        I/O ERROR DETECTED - ANALYSE THE SITUATION
*-------------------------------------------------------------------
*
* EOT IS AN 'ACCEPTABLE' CONDITION, INDICATED BY UNIT EXCEPTION
         TM    IOBCSW+3,X'Ø1'            * UNIT EXCEPTION
         BO    ERROR8                    * YES - MEANS END OF TAPE
* UNIT CHECK PROBABLY MEANS HARD ERROR
         TM    IOBCSW+3,X'Ø2'            * UNIT CHECK?
         BNO   ERROR2Ø                   * THIS SHOULD NOT HAPPEN
         TM    IOBSENSØ,X'8Ø'            * COMMAND REJECT?
         BO    ERROR12                   * YES - PROBABLY WRITE-PROTECT
```

© 1999. Reproduction prohibited. Please inform Xephon of any infringement.

```
         TM    IOBSENSØ,X'Ø8'           * DATA CHECK?
         BNO   ERROR2Ø                  * NO - OTHER HARD ERROR
         L     R6,Ø(R3)                 * CCW COUNT
         SR    R8,R8                    * R8 = ...
         ICM   R8,B'ØØ11',IOBCSW+5      * ... RESIDUAL COUNT
         SR    R6,R8                    * PASS # BYTES WRITTEN ...
         ST    R6,Ø(R3)                 * ... TO CALLER, AND ...
         B     ERROR16                  *     ... INDICATE DATA CHECK
         DROP  R1Ø                      * FINISHED WITH IOB
         EJECT
*
*********************************************************************
*        CLOSE THE TAPE
*********************************************************************
*
CLOSE    EQU   *
         L     R11,ADCB                 * R11 = DCB ADDRESS
         CLOSE ((R11)),MODE=31          * CLOSE DCB
         SR    R15,R15                  * CLEAR RETURN CODE
         EJECT
*********************************************************************
*        RETURN TO CALLER
*********************************************************************
RETURN   EQU   *
         ST    R15,Ø(R5)                * STORE IOSTAT FOR CALLER
         L     R13,4(R13)               * RESTORE ADDRESS OF HSA
         L     R14,12(R13)              * RESTORE R14
         LM    RØ,R12,2Ø(R13)           * RESTORE RØ-R12
         BR    R14                      * AND RETURN
         EJECT
*********************************************************************
*        ERROR CONDITIONS
*********************************************************************
ERROR4   EQU   *
         LA    R15,4                    * OPEN FAILURE
         B     RETURN                   * RETURN
ERROR8   EQU   *
         LA    R15,8                    * END OF TAPE
         B     RETURN                   * RETURN
ERROR12  EQU   *
         LA    R15,12                   * WRITE PROTECTED
         B     RETURN                   * RETURN
ERROR16  EQU   *
         LA    R15,16                   * DATA CHECK
         B     RETURN                   * RETURN
ERROR2Ø  EQU   *
         LA    R15,2Ø                   * OTHER HARD ERROR
         B     RETURN                   * RETURN
         EJECT
*********************************************************************
*        CONSTANTS, VARIABLES AND DATA AREAS
```


© 1999. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (940) 455 7050, fax (940) 455 2492.

```
*************************************************************************
         DS    ØD
         DC    CL8'SAVEAREA'
SAVEAREA DS    18F
ADCB     DC    A(DCB)
ADEB     DC    A(Ø)
AIOB     DC    A(IOBAREA)
ACHPROG  DC    A(CHPROG)
         DS    ØF
DCB      DCB   DDNAME=DUMMY,DSORG=PS,DEVD=TA,MACRF=E
LDCB     EQU   *-DCB
         DS    ØF
IOBAREA  DC    (LIOB)X'ØØ'
         DS    ØD
CHPROG   EQU   *
CCW1     CCW   X'Ø1',IDAW,X'64',Ø        * WRITE
CCW2     CCW   X'Ø3',Ø,Ø,1               * NOP
IDAW     DS    F                         * INDIRECT DATA ADDRESS WORD
LCHPROG  EQU   *-CHPROG
W        CCW   X'Ø1',IDAW,X'64',Ø        * WRITE
WTM      CCW   X'1F',Ø,X'44',1           * WRITE TAPE MARK
IOB      DSECT
IOBFLAG1 DS    XL1                       * CHAINING/UNRELATED BITS
IOBFLAG2 DS    XL1                       * NOT USED HERE
IOBSENSE DS    ØXL2                      * SENSE BYTES
IOBSENSØ DS    XL1                       *   SENSE BYTE 1
IOBSENS1 DS    XL1                       *   SENSE BYTE 2
IOBECBCC DS    XL1                       * FIRST BYTE OF COMP. CODE
IOBECBPT DS    AL3                       * ECB ADDRESS
IOBFLAG3 DS    XL1                       * SYSTEM USE ONLY
IOBCSW   DS    XL7                       * CHANNEL STATUS WORD
IOBSIOCC DS    XL1                       * START SUBCHANNEL DATA
IOBSTART DS    AL3                       * CHANNEL PROGRAM ADDRESS
         DS    XL1                       * RESERVED
IOBDCBPT DS    AL3                       * DCB ADDRESS
IOBRESTR DS    XL1                       * USED FOR ERROR RECOVERY
         DS    XL3                       * USED FOR ERROR RECOVERY
IOBINCAM DS    XL2                       * USED FOR MAG TAPE ONLY
         DS    XL2                       * RESERVED
IOBECB   DS    F                         * ECB
LIOB     EQU   *-IOB
         PRINT NOGEN
         DCBD  DSORG=PS,DEVD=TA          * DCB MAPPING MACRO
         END
```

P R S Wright
Associate Consultant
Tessella Support Services (UK)
©Xephon 1999

© 1999. Reproduction prohibited. Please inform Xephon of any infringement.

The binder application interface

INTRODUCTION

DFSMS/MVS (program management) provides the binder to replace the linkage editor from previous versions of MVS. The binder (and new loader) overcome some of the restrictions that have always been part of the linkage editor and the batch loader. This article will look at the differences between the binder and the linkage editor and some of the benefits that can be obtained by using the binder. We will also look at an example of using the application programming interface provided with the binder. This interface enables the caller to communicate directly with the binder from a batch environment. It can be used to dynamically bind program objects or obtain information on objects and also on load modules.

The main difference between the two products is in the output they deliver. The linkage editor generates what is known as a load module by using object code (the output from a compiler) and existing load modules to generate a new load module. The load module is saved into a PDS and can be executed through JCL, a LINK macro, or a CALL. In addition to supporting most linkage editor functions, the binder can also generate what is known as program objects. (Input to the binder can consist of object code, load modules, and other program objects.) Program objects extend the functions of load modules. They are stored into PDSE or HFS files and have a one gigabyte size restriction, as opposed to the 16MB restriction of load modules. By storing program objects into a PDSE rather than load modules into a PDS, the restriction of a maximum of 32,767 external names in a PDS is overcome, the only restriction now is the actual size of the PDSE.

So how do you invoke the binder? Exactly the same as the linkage editor. The program name is still IEWL. By looking at the type of output dataset involved, the binder decides whether to generate a load module (PDS) or a program object (HFS and PDS/E, also known as a library). The result of the decision is listed in the output from the binder. The following shows the difference in output received during link/bind time between linking or binding the same object into a PDS and a PDS/E:

© 1999. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (940) 455 7050, fax (940) 455 2492.

```
SAVE OPERATION SUMMARY:
   PROGRAM TYPE    LOAD MODULE

SAVE OPERATION SUMMARY:
   PROGRAM TYPE    PROGRAM OBJECT(FORMAT 2)
```

You can use IEBCOPY to convert load modules to program objects simply by copying a PDS to a PDSE. The opposite is also true: a program object can be converted back to a load module, provided that no load module restriction is exceeded. It is possible that later versions of language products may have a dependency on binder functions, so a gradual movement towards program objects should be considered. With current versions of MVS, both the linkage editor and binder are still available and in (rare) cases where the linkage editor is required, it can be invoked by using entry points HEWLKED or HEWLF064, the batch loader can be invoked by using entry point HEWLDIA.

There are more differences between load modules and program objects. We will look at one that causes some inconvenience to systems programmers. We will then use the application interface to develop a utility to overcome this ‘problem’.

THE PROBLEM

With a load module we can ‘IEAEYEBALL’ a load module to see the link date. We use the browse option under ISPF, enter HEX ON and an experienced eye will then know where to look for the date. Here is an example:

```
————————————————————————————————
. ......TEMP    .......
28000001ECDD444400000004
00000100354700000000002000
————————————————————————————————-

 ..5695DF108 ..q..
810FFFFCCFFF400921
01256954610801187F
 ————————————————————————————————-
```

From this we can see that the module was linked on 98271 (the Julian date). This is evident from the rightmost bytes on the second line in the example.

Doing the same for a program object delivers no usable information:

© 1999. Reproduction prohibited. Please inform Xephon of any infringement.

```
IEWPLMH ... ..................... ........... ...
CCEDDDC4000700000001000000000000700010000000800O
9567348000081000000000080100000800040200000C000
```

(Note the IEWPLMH in the first 8 bytes – this is always the case in a program object.)

This program will give you the ability to see the date the program object was bound. It is also usable with load modules. It is called by entering 'POBJINFO modname' from TSO, following which the user will be prompted for the name of the dataset the module is in. A REXX routine is supplied to allocate the dataset and call the utility. The result is displayed in the format:

```
Linked on 1998274 at 12:04:13 by JOBNAME
```

(Note that the REXX also allocates a file by the name of IEWINFO. To see the informational messages delivered by the binder during execution, make a change to the Assembler source at the FILENAME label as indicated by the documentation.)

REXX POBJL

```
/* REXX */
arg mem
do while (mem="")
   say "Enter the member name"
   pull mem
end
modlib=""
do while (modlib="")
   say "Enter the name of the LIBRARY/PDS to scan"
   pull modlib
end
if sysdsn("'"modlib"("mem")'") = "OK" then do
    "alloc fi(IEWINFO) da(*) shr reuse"
    "alloc fi(IEWLIB) da('"modlib"') shr reuse"
    "call 'YOUR.LINKLIB(pobjinfo)' '"mem"'"
    "free fi(IEWLIB)"
    "free fi(IEWINFO)"
    end
else say sysdsn("'"modlib"("mem")'")
```


© 1999. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (940) 455 7050, fax (940) 455 2492.

ASSEMBLER SOURCE

```
POBJINFO CSECT
POBJINFO AMODE 31
POBJINFO RMODE 24
         BAKR  R14,Ø                Save caller's Status
         LR    R12,R15              Pick up our load address
         USING POBJINFO,R12
*********************************************************************
*        Main routine
*********************************************************************
START    LA    R3,STORSIZE          Size of storage to get and clear
         LR    R5,R1                Preserve passed pointer
         STORAGE OBTAIN,LENGTH=(3),LOC=ANY
         LR    R2,R1                Address of obtained area
         LA    R3,STORSIZE          Length of the area
         XR    R9,R9                Byte to propagate
         MVCL  R2,R8                Propagate binary zeroes
         USING STORAREA,R1
         ST    R13,SAVEAREA+4       Back chain
         DROP  R1
         LR    R13,R1               Address of obtained area
         USING STORAREA,R13         Addressability to obtained area
         LR    R1,R5                Restore passed pointer
         BAS   R14,CHECKPRM         Make sure module name passed
         LTR   R15,R15              Successful?
         BNZ   RETURN               No, get out
         BAS   R14,INITBUFF         Go obtain and initialize buffer
         LTR   R15,R15              Successful?
         BNZ   RETURN               No, get out
         BAS   R14,STRTDIAG         Go start the binder dialog
         LTR   R15,R15              Successful?
         BNZ   RETURN               No, get out
         BAS   R14,CRTWMOD          Go create a workmod
         LTR   R15,R15              Successful?
         BNZ   CLEANUP2             No, get out
         BAS   R14,SETOPT           Go set the LIST option to ALL
         LTR   R15,R15              Successful?
         BNZ   CLEANUP1             No, get out
         BAS   R14,INCLMOD          Go INCLUDE the module
         LTR   R15,R15              Successful?
         BNZ   CLEANUP1             No, get out
         BAS   R14,STORDATA         Go get the required data
         LTR   R15,R15              Successful?
         BNZ   CLEANUP1             No, get out
         TPUT  DATETIME,L'DATETIME  Info we wanted
CLEANUP1 BAS   R14,DELWMOD          Go delete the workmod
CLEANUP2 BAS   R14,ENDDIAG          Go end the dialog
RETURN   L     R4,RETCODE           Pick up return code
         LR    R2,R13               Pointer to storage area
         LA    R3,STORSIZE          Size of storage to free
```

© 1999. Reproduction prohibited. Please inform Xephon of any infringement.

```
         STORAGE RELEASE,LENGTH=(3),ADDR=(2)
         LR    R15,R4              Reload return code
         PR                        Back to our caller
*************************************************************************
*        This routine picks up the passed module name
*************************************************************************
CHECKPRM BAKR  R14,Ø                Store caller's mode
         L     R1,Ø(R1)            Point to passed parm
         CLC   Ø(2,R1),=H'1'       Name must be at least 1 byte long
         BL    INVLPARM            Invalid parm passed
         CLC   Ø(2,R1),=H'8'       Must not be longer than 8 bytes
         BH    INVLPARM            Parm too long
         LH    R2,Ø(R1)            Pick up the member name length
         BCTR  R2,Ø                Correct the length
         EX    R2,MVCNAME          Execute the MVC instruction
         XR    R15,R15             Clear return code
         B     CHECKPRX            Get out
MVCNAME  MVC   MEMNAME+2(Ø),2(R1)   Move parm into member name field
INVLPARM TPUT  INVLP,L'INVLP       Give message
         LA    R15,4               Set return code
CHECKPRX PR                        Back to our caller
*************************************************************************
*        This routine obtains and initializes the buffer
*************************************************************************
INITBUFF BAKR  R14,Ø                Store caller's mode
         IEWBUFF FUNC=GETBUF,TYPE=IDRB
         LTR   R15,R15             Successful?
         BNZ   NOBUFF              No, failed
         IEWBUFF FUNC=INITBUF,TYPE=IDRB
         ST    R6,HEADER@          Preserve the header address
         ST    R7,ENTRY@           Preserve the entry address
         LTR   R15,R15             Successful?
         BZ    INITBUFX            Yes, get out
NOBUFF   LR    R2,R15              Preserve return code
         ST    R15,RETCODE         Store the return code
         WTO   'Failed to obtain and init work buffer',               X
               ROUTCDE=11
         LR    R15,R2              Reload return code
INITBUFX PR                        Back to our caller
*************************************************************************
*        This routine starts the binder dialog
*************************************************************************
STRTDIAG BAKR  R14,Ø                Store caller's mode
         IEWBIND FUNC=STARTD,DIALOG=DTOKEN,FILES=FILENAME,           X
               RETCODE=RETCODE,RSNCODE=RSNCODE
         L     R15,RETCODE         Look at the return code
         LTR   R15,R15             Successful?
         BZ    STRTDIAX            Yes, get out
NOSTART  LR    R2,R15              Preserve return code
         TPUT  NODIAG,L'NODIAG     Give "no dialog" message
```


© 1999. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (940) 455 7050, fax (940) 455 2492.

```
         BAS   R14,SHOWCODE         Go print return and reason codes
         LR    R15,R2              Reload return code
STRTDIAX PR                        Back to our caller
*********************************************************************
*        This routine creates a workmod with ACCESS intent
*********************************************************************
CRTWMOD  BAKR  R14,Ø                Store caller's mode
         IEWBIND FUNC=CREATEW,DIALOG=DTOKEN,                           X
               INTENT=ACCESS,WORKMOD=WTOKEN,                           X
               RETCODE=RETCODE,RSNCODE=RSNCODE
         L     R15,RETCODE          Look at the return code
         LTR   R15,R15              Successful?
         BZ    CRTWMODX             Yes, get out
NOCREATE LR    R2,R15               Preserve return code
         TPUT  NOWMOD,L'NOWMOD      Give "no workmod" message
         BAS   R14,SHOWCODE         Go print return and reason codes
         LR    R15,R2               Reload return code
CRTWMODX PR                        Back to our caller
*********************************************************************
*        This routine sets the LIST option to SUMMARY
*********************************************************************
SETOPT   BAKR  R14,Ø                Store caller's mode
         IEWBIND FUNC=SETO,OPTION=OPTNLIST,                            X
               WORKMOD=WTOKEN,OPTVAL=SUMMARY,                          X
               RETCODE=RETCODE,RSNCODE=RSNCODE
         L     R15,RETCODE          Look at the return code
         LTR   R15,R15              Successful?
         BZ    SETOPTX              Yes, get out
NOSETOPT LR    R2,R15               Preserve return code
         TPUT  NOOPT,L'NOOPT        Give "not set" message
         BAS   R14,SHOWCODE         Go print return and reason codes
         LR    R15,R2               Reload return code
SETOPTX  PR                        Back to our caller
*********************************************************************
*        This routine INCLUDEs the module
*********************************************************************
INCLMOD  BAKR  R14,Ø                Store caller's mode
         IEWBIND FUNC=INCLUDE,WORKMOD=WTOKEN,                          X
               DDNAME=INCLLIB,MEMBER=MEMNAME,INTYPE=NAME,              X
               RETCODE=RETCODE,RSNCODE=RSNCODE
         L     R15,RETCODE          Look at the return code
         LTR   R15,R15              Successful?
         BZ    SETOPTX              Yes, get out
NOINCLUD LR    R2,R15               Preserve return code
         TPUT  NOINCL,L'NOINCL      Give "not included' message
         BAS   R14,SHOWCODE         Go print return and reason codes
         LR    R15,R2               Reload return code
INCLMODX PR                        Back to our caller
*********************************************************************
*        This routine gets the required data
*********************************************************************
```

© 1999. Reproduction prohibited. Please inform Xephon of any infringement.

```
STORDATA BAKR  R14,Ø               Store caller's mode
         L     R6,HEADER@          Reload the header address
         L     R7,ENTRY@           Reload the buffer address
         IEWBIND FUNC=GETD,WORKMOD=WTOKEN,AREA=IEWBIDB,                X
               CURSOR=NULL,COUNT=NUMBYTES,CLASS=CLASS,                 X
               RETCODE=RETCODE,RSNCODE=RSNCODE
         L     R15,RETCODE         Look at the return code
         CH    R15,=H'4'           Successful?
         BH    NOGETDTA            No
MOVEDATE MVC   DATETIME+1Ø(7),IDB_DATE_BOUND   Date we wanted
         CLC   IDB_TIME_BOUND,=6C'ØØ'          Time available?
         BNE   MOVETIME                        No
         MVC   DATETIME+18(11),=11X'4Ø'        Date is not available
         B     MOVECALL
MOVETIME MVC   DATETIME+21(2),IDB_TIME_BOUND   First part of time
         MVC   DATETIME+24(2),IDB_TIME_BOUND+2 Second part of time
         MVC   DATETIME+27(2),IDB_TIME_BOUND+4 Third part of time
MOVECALL CLC   IDB_CALLERID_CHARS,=H'28'       Job info available?
         BNL   MOVEJBNM            Move the job name
         MVC   DATETIME+3Ø(11),=11X'4Ø' Info not available
         B     CLEARR15
MOVEJBNM MVC   DATETIME+33(8),IDB_CALLERID+2Ø
CLEARR15 XR    R15,R15             Data successfully obtained
         B     STORDATX            Get out
NOGETDTA LR    R2,R15              Preserve return code
         TPUT  NODATA,L'NODATA     Give "no data" message
         BAS   R14,SHOWCODE        Go print return and reason codes
         LR    R15,R2              Reload return code
STORDATX PR                        Back to our caller
*********************************************************************
*        This routine deletes the workmod
*********************************************************************
DELWMOD  BAKR  R14,Ø               Store caller's mode
         IEWBIND FUNC=DELETEW,WORKMOD=WTOKEN,                          X
               RETCODE=RETCODE,RSNCODE=RSNCODE
         L     R15,RETCODE         Look at the return code
         LTR   R15,R15             Successful?
         BZ    DELWMODX            Yes, get out
         LR    R2,R15              Preserve return code
         TPUT  NODEL,L'NODEL       Give "not deleted" message
         BAS   R14,SHOWCODE        Go print return and reason codes
         LR    R15,R2              Reload return code
DELWMODX PR                        Back to our caller
*********************************************************************
*        This routine ends the dialog
*********************************************************************
ENDDIAG  BAKR  R14,Ø               Store caller's mode
         IEWBIND FUNC=ENDD,DIALOG=DTOKEN,                              X
               RETCODE=RETCODE,RSNCODE=RSNCODE
         L     R15,RETCODE         Look at the return code
         LTR   R15,R15             Successful?
```


© 1999. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (940) 455 7050, fax (940) 455 2492.

```
         BZ    ENDDIAGX             Yes, get out
         LR    R2,R15               Preserve return code
         TPUT  NOEND,L'NOEND        Give "not ended" message
         BAS   R14,SHOWCODE         Go print return and reason codes
         LR    R15,R2               Reload return code
ENDDIAGX PR                         Back to our caller
*********************************************************************
*        This routine makes the return and reason codes printable
*********************************************************************
SHOWCODE BAKR  R14,Ø                Store caller's mode
         MVC   LOWBYTES,RETCODE     Move return and reason code in
         NC    LOWBYTES(8),=8X'FØ'  Turn off the second part bytes
         TR    LOWBYTES(8),LEFTHALF
         MVC   HIGBYTES,RETCODE     Move return and reason code in
         NC    HIGBYTES(8),=8X'ØF'  Turn off the first part bytes
         TR    HIGBYTES(8),RIGTHALF
         LA    R1,LOWBYTES          Where the first half of each byte is
         LA    R2,HIGBYTES          Where second half of each byte is
         LA    R3,WORKAREA          Where we want to move the data to
         LA    R4,8                 8 bytes to move
CODELOOP MVC   Ø(1,R3),Ø(R1)        Move first half of byte
         LA    R3,1(R3)             Bump up target pointer
         MVC   Ø(1,R3),Ø(R2)        Move second half of byte
         LA    R3,1(R3)             Bump up target pointer
         LA    R1,1(R1)             Bump up first-half-of-byte pointer
         LA    R2,1(R2)             Bump up second-half-of-byte pointer
         BCT   R4,CODELOOP          Do for each of the bytes
         MVC   CODEMSG+12(8),WORKARea
         MVC   CODEMSG+34(8),WORKARea+8
CODEWTO  TPUT  CODEMSG,L'CODEMSG
SHOWCODX PR
*********************************************************************
*        Constants follow
*********************************************************************
         LTORG
INVLP    DC    C'Invalid member name passed.'
NODIAG   DC    C'Failed to START dialog.'
NOWMOD   DC    C'Failed to CREATE a workmod.'
NOOPT    DC    C'Failed to set LIST=SUMMARY option.'
NOINCL   DC    C'Failed to INCLUDE module.'
NODATA   DC    C'Failed to obtain data for module.'
NODEL    DC    C'Failed to DELETE workmod.'
NOEND    DC    C'Failed to END dialog.'
MEMNAME  DC    H'8',CL8' '
INCLLIB  DC    H'6',C'IEWLIB'
CLASS    DC    H'6',C'B_IDRB'
CODEMSG  DC    C'Return CODE=xxxxxxxx, reason code=xxxxxxxx.'
         IEWBUFF FUNC=MAPBUF,TYPE=IDRB,SIZE=1ØØØ,HEADREG=6,ENTRYREG=7
FILENAME DS    ØF
         DC    F'Ø'                  Swap with next card to get BINDER
*        DC    F'2'                    messages displayed
```

© 1999. Reproduction prohibited. Please inform Xephon of any infringement.

```
         DC    CL8'TERM',F'8',A(TERM)
         DC    CL8'PRINT',F'8',A(TERM)
TERM     DC    CL7'IEWINFO'          DD-name messages will go to
OPTNLIST DC    H'4',C'LIST'          Option LIST=SUMMARY
SUMMARY  DC    H'7',C'SUMMARY'       Option LIST=SUMMARY
DATETIME DC    C'Linked on xxxxxxx at yy:yy:yy by zzzzzzzz'
LEFTHALF DS    ØCL24Ø
         DC    X'FØ',15X'ØØ',X'F1',15X'ØØ',X'F2',15X'ØØ',X'F3'
         DC    15X'ØØ',X'F4',15X'ØØ',X'F5',15X'ØØ',X'F6',15X'ØØ',X'F7'
         DC    15X'ØØ',X'F8',15X'ØØ',X'F9',15X'ØØ',X'C1',15X'ØØ',X'C2'
         DC    15X'ØØ',X'C3',15X'ØØ',X'C4',15X'ØØ',X'C5',15X'ØØ',X'C6'
RIGTHALF DC    X'FØF1F2F3F4F5F6F7F8F9C1C2C3C4C5C6'
*********************************************************************
*        DSECTS follow
*********************************************************************
STORAREA DSECT
SAVEAREA DS    18F
DTOKEN   DS    D                     Dialog token
WTOKEN   DS    D                     Workmod token
DOUBLE   DS    D                     General workarea
RETCODE  DS    F                     Return code
RSNCODE  DS    F                     Reason code
LOWBYTES DS    CL8                   Workarea to make codes printable
HIGBYTES DS    CL8                   Workarea to make codes printable
WORKAREA DS    CL16                  Print format return and reason
codes
NULL     DS    F                     Where binder should begin
NUMBYTES DS    F                     Number of bytes returned to us
HEADER@  DS    F                     Address of IEW header
ENTRY@   DS    F                     Address of IEW data entry
STORSIZE EQU   *-STORAREA
*
R1       EQU   1                     Register equates
R2       EQU   2
R3       EQU   3
R4       EQU   4
R5       EQU   5
R6       EQU   6
R7       EQU   7
R8       EQU   8
R9       EQU   9
R1Ø      EQU   1Ø
R11      EQU   11
R12      EQU   12
R13      EQU   13
R14      EQU   14
R15      EQU   15
         END
```

Gerty Brits
Mindalore Consulting (India)
© Xephon 1999

© 1999. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (940) 455 7050, fax (940) 455 2492.

An ISPF search facility

INTRODUCTION

The standard search facility under ISPF, via option 3.14, is very useful but it has two slight flaws:

- The user has to remember and type in the dataset name that needs to be searched.
- The search facility produces a report with members and their lines containing the search argument. If a user wishes to access these members, they also have to be remembered.

To reduce the need to remember and type commands, I wrote a search program in REXX that can be executed from ISPF option 3.4 as a line command. I have called it XF and it uses the standard ISPF super compare program ISRSUPC. The syntax is: XF 'search argument' or XF / 'search argument' for people used to that notation. It searches the PDS and displays a member list with hits that can be edited or viewed.

XF REXX

```
/* REXX; FIND MEMBERS IN A PDS WITH SEARCH ARGUMENT IN ISPF =3.4 */
/* XF=EXTRA FIND; DD  IS DATASET NAME, NOT ENTERED IN =3.4       */
/*                SE1 IS SEARCH KEYWORD                          */
ARG DD SE1
/* CHECK INPUT PARAMETERS */
CALL MSG(OFF)
E=LISTDSI(DD)
IF SYSREASON > Ø THEN
   DO
   E=LISTDSI(SE1)
   IF SYSREASON > Ø THEN
      DO
      ZEDSMSG = 'NO VALID DATASET'
      ZEDLMSG = 'AND ONLY 1 SEARCH ARGUMENT ALLOWED'
      "ISPEXEC SETMSG MSG(ISRZØØ1)"
      EXIT
      END
      ELSE
      DO
      SE2 = SE1
      SE1 = DD
      DD  = SE2
      END
```

© 1999. Reproduction prohibited. Please inform Xephon of any infringement.

```
   END
DSNAME=SUBSTR(DD,2,LENGTH(DD)-2)
IF SE1 = '' THEN DO
    ZEDSMSG = 'NO SEARCH ARGUMENT'
    ZEDLMSG = 'USE A SEARCH ARGUMENT'
    "ISPEXEC SETMSG MSG(ISRZØØ1)"
    EXIT
END
/* ALLOCATE THE NESSECARY DATASETS FOR ISPF STANDARD SEARCH PROGRAM */
ADDRESS TSO
'FREE FI(NEWDD,OUTDD,SYSIN)'
"ALLOC FI(NEWDD) SHR DA('"DSNAME"')"
'ALLOC FI(OUTDD) NEW DSORG(PS) REC(F B) LR(133) BLK(133ØØ)',
     'SPACE(2,2) TRACKS DA(XF.LIJST)'
'ALLOC FI(SYSIN) DELETE DSORG(PS) REC(F B) LR(8Ø) BLK(312Ø)',
     'SPACE(1,2) TRACKS'
QUEUE 'SRCHFOR' "'"SE1"'"
'EXECIO 1 DISKW SYSIN (FINIS'
/* ISSUE THE ISPF SEARCH */
ADDRESS ISPEXEC
'ISPEXEC SELECT PGM(ISRSUPC) PARM(SRCHCMP,ANYC,NOSEQ,LMTO)'
/* FREE THE DATASETS AND READ THE RESULTS INTO A BUFFER */
ADDRESS TSO
'FREE FI(NEWDD,OUTDD,SYSIN)'
'ALLOC FI(OUTDD) SHR DA(XF.LIJST) DELETE'
'EXECIO * DISKR OUTDD (FINIS'
'FREE FI(OUTDD)'
/* READ THE BUFFER AND PUT VALID MEMBERS INTO A TABLE */
N=Ø
MEMBER.=''
DO QUEUED()
   PULL REGEL
   IF SUBSTR(REGEL,2,1) = ' ' THEN ITERATE
   IF SUBSTR(REGEL,2,11) = 'LINES-FOUND' THEN ITERATE
   IF SUBSTR(REGEL,2,11) = 'MEMBER-SEAR' THEN ITERATE
   IF SUBSTR(REGEL,2,11) = 'PROCESS OPT' THEN ITERATE
   IF SUBSTR(REGEL,2,11) = 'THE FOLLOWI' THEN ITERATE
   N=N+1
   MEMBER.N = SUBSTR(REGEL,2,9)
END
/* CHECK IF THERE ARE MEMBERS FOUND */
IF MEMBER.1='' THEN DO
   ZEDSMSG = 'NOTHING FOUND'
   ZEDLMSG = 'THERE ARE NO MEMBERS WITH' SE1
   "ISPEXEC SETMSG MSG(ISRZØØ1)"
   EXIT
END
/* PUT THE TABLE INTO A ISPF TABLE */
ADDRESS ISPEXEC
'TBCREATE MEMSEL NAMES(MEMBER) NOWRITE REPLACE'
DO X=1 TO 99999
   IF MEMBER.X='' THEN LEAVE
   MEMBER=STRIP(MEMBER.X)
```


© 1999. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (940) 455 7050, fax (940) 455 2492.

```
   'TBADD MEMSEL'
END
/* DISPLAY THE MEMBERLIST */
'TBTOP MEMSEL'
'ADDPOP ROW(1) COLUMN(9)'
'TBDISPL MEMSEL PANEL(XFPANEL)'
PANEL_ACTION:
IF REPLY='END' THEN EXIT
IF ZTDSELS=Ø THEN 'TBDISPL MEMSEL'
IF ZTDSELS=1 THEN DO
    CONTROL DISPLAY SAVE
    IF T = 'E' THEN
       "EDIT DATASET('"DSNAME"("MEMBER")')"
       ELSE
       "VIEW DATASET('"DSNAME"("MEMBER")')"
    CONTROL DISPLAY RESTORE
   'TBDISPL MEMSEL'
END
IF ZTDSELS>1 THEN DO UNTIL ZTDSELS=1
   MEMBER=STRIP(MEMBER)
    CONTROL DISPLAY SAVE
    IF T = 'E' THEN
       "EDIT DATASET('"DSNAME"("MEMBER")')"
       ELSE
       "VIEW DATASET('"DSNAME"("MEMBER")')"
    CONTROL DISPLAY RESTORE
   'TBDISPL MEMSEL'
END
SIGNAL PANEL_ACTION
```

XFPANEL

```
)attr default(%+_)
! type(output) intens(high) caps(on) just(left)
$ type(input) intens(low) caps(on) just(asis)
)body window(76,19)
 %command ===> _zcmd                          %scroll ===>_amt +
 +
 member
)model
$t!z       +
)init
.zvars ='(member)'
&amt = page
)reinit
&t=''
)PROC
&reply=.resp
)END
```

Willie van Tilburg
Systems Programmer (The Netherlands)
© Xephon 1999

© 1999. Reproduction prohibited. Please inform Xephon of any infringement.

PDS member change management detection

INTRODUCTION

In *MVS Update* Issue 112 (January 1996), I published an Assembler program called CRC32, which calculates Cyclic Redundancy Check (CRC) values for various types of dataset that can then be used for file verification purposes. One of the benefits of this utility was its ability to develop CRC values for each member of a partitioned dataset. Using these individual member CRC values, I developed a process that allows the detection of updates to members based on changes in their CRC values. I instituted this system at my current site to perform daily change checking on critical MVS system datasets. This was done because these MVS system datasets were not under control of our current change control package (Endevor), and even though they may or may not be controlled by SMP/E, there was still the ability to directly update them. The typical instance is the addition or change of members in SYS1.PARMLIB, for example, which, although managed by SMP/E in general, is regularly updated by individuals. This is especially true since many members in SYS1.PARMLIB can have their changes introduced into the MVS operating system via the MVS SET operator command.

There are two REXX EXECs involved in the process. The CRCVER EXEC builds the CRC values for a given dataset by running the CRC32 command against the specified dataset, optionally on a specified volume, to build the member CRC values, then edits and stores the resulting list as a member in another PDS (that I will refer to as the CRC database), which must have been pre-allocated with the DCB attributes of DSORG=PO, LRECL=120, and RECFM=FB. The member name is in the format of D/yy/mm/dd where yy/mm/dd is the year, month and day date that the process took place on. The CRC database names (there will be one for each PDS that is being checked) also have a special format. I have arbitrarily used a high level of CRC. The second level is Vvvvvvv where vvvvvv is the volume serial that the dataset whose CRC is being calculated resides. This convention is necessary to handle the checking of different systems residence volumes which contain like named datasets, for example, to

 © 1999. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (940) 455 7050, fax (940) 455 2492.

differentiate SYS1.PARMLIB on volume SYSRSA versus volume SYSRSB. The remaining qualifiers correspond exactly to the name of the dataset whose CRC values are being calculated. Thus, to perform this process for SYS1.PARMLIB on both SYSRSA and SYSRSB, the following CRC databases would be pre-allocated with the attributes specified above:

- CRC.VSYSRSA.SYS1.PARMLIB
- CRC.VSYSRSB.SYS1.PARMLIB .

For datasets that are unique, the same CRC database name format must still be followed, but the volume serial that the dataset resides on will be automatically determined by the fact that the CRCVER EXEC is not passed a second parameter specifying a volume serial.

The second EXEC, CRCTRAP, uses the TSO LISTCAT command to list all datasets under high-level qualifier CRC. In this way, when any new CRC databases are built, they will be automatically scanned. For each CRC database found, the TSO LISTDS command is issued to get a list of all member names that were generated by the CRCVER process. Since the member names are in order, the last two member names contain the two most recent results of the CRCVER process. The CRCTRAP EXEC compares these two most recent members, using the ISPF SuperC compare utility to determine if there have been any changes in the PDS since the previous day's process. This process occurs for each CRC database. As a consequence of this process, on the first day this process runs, you will need just to run the CRCVER processes against each target dataset, without running the CRCTRAP EXEC.

Using the ISRSUPC output, you can then determine if a member has been changed because the CRC values are different between the two most recent days as shown on the pairs of I and D ID lines in the sample output shown below for SYS1.VTAMLST. You can also determine if members were either added or deleted from the target dataset, by the appearance of a single I or D ID line, as shown below for SYS1.LINKLIB(MERKNOW). For further understanding of either the CRC32 command or the output from the ISRSUPC utility, you should refer to *MVS Update* Issue 112 where CRC32 appeared, and the appropriate IBM ISPF manuals, respectively.

© 1999. Reproduction prohibited. Please inform Xephon of any infringement.

```
SUPERC - MVS/PDF FILE/LINE/WORD/BYTE/SFOR COMPARE UTILITY - V4.20(ISPF)
NEW: CRC.VSYSRSA.SYS1.VTAMLST(D980813)
OLD: CRC.VSYSRSA.SYS1.VTALMST(D980812)

                    LISTING OUTPUT SECTION (LINE COMPARE)

ID       SOURCE LINES

---+---1---+---2---+---3---+---4---+---5---+---6---+---7---+

I - 19980813 06:08:00 member    ADJSSIIN FBD91C5C SYSRSA SYS1.VTAMLST
D - 19980812 06:06:03 member    ADJSSIIN 0F9FB427 SYSRSA SYS1.VTAMLST

I - 19980813 06:08:00 member    ATCCON14 45E3B0F8 SYSRSA SYS1.VTAMLST
D - 19980812 06:06:03 member    ATCCON14 9BBE924E SYSRSA SYS1.VTAMLST

I - 19980813 06:08:00 directory          24E6CCA5 SYSRSA SYS1.VTAMLST
D - 19980812 06:06:03 directory          64D24996 SYSRSA SYS1.VTAMLST
I - 19980813 06:08:00 members            581C9592 SYSRSA SYS1.VTAMLST
D - 19980812 06:06:03 members            C034BFB3 SYSRSA SYS1.VTAMLST
I - 19980813 06:08:00 dataset            8305A6C8 SYSRSA SYS1.VTAMLST
D - 19980812 06:06:03 dataset            5B1909DA SYSRSA SYS1.VTAMLST

  SUPERC - MVS/PDF FILE/LINE/WORD/BYTE/SFOR COMPARE UTILITY - V4.20(ISPF)
NEW: CRC.VSYSRSB.SYS1.LINKLIB(D980813)   OLD:
CRC.VSYSRSB.SYS1.LINKLIB(D980812)

                    LISTING OUTPUT SECTION (LINE COMPARE)

ID       SOURCE LINES

---+---1---+---2---+---3---+---4---+---5---+---6---+---7---+

I - 19980813 06:06:04 member    MERKNOW  79D4D731 SYSRSB SYS1.LINKLIB

I - 19980813 06:06:04 directory          1D162E20 SYSRSB SYS1.LINKLIB
D - 19980812 06:02:24 directory          ECBEEE58 SYSRSB SYS1.LINKLIB
I - 19980813 06:06:04 members            39C106DE SYSRSB SYS1.LINKLIB
D - 19980812 06:02:24 members            BFEA2E10 SYSRSB SYS1.LINKLIB
I - 19980813 06:06:04 dataset            DB28D701 SYSRSB SYS1.LINKLIB
D - 19980812 06:02:24 dataset            ACAB3FB7 SYSRSB SYS1.LINKLIB
```

Figure 1: Sample ISRSPUC output

An example of a sample JCL for PDS member CRC generation and checking is shown below.

© 1999. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (940) 455 7050, fax (940) 455 2492.

SAMPLE JCL

```
//TSO      EXEC PGM=IKJEFTØ1,DYNAMNBR=5Ø,REGION=16M
//SYSPROC  DD DISP=SHR,DSN=user.clist.rexx.library
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  *
%CRCVER 'SYS1.LINKLIB' SYSRSA
%CRCVER 'SYS1.PARMLIB' SYSRSA
%CRCVER 'SYS1.PROCLIB' SYSRSA
%CRCVER 'SYS1.LINKLIB' SYSRSB
%CRCVER 'SYS1.PARMLIB' SYSRSB
%CRCVER 'SYS1.PROCLIB' SYSRSB
%CRCVER 'SYSØ.PPLIB'
%CRCVER 'SYSØ.STCJOBS'
%CRCVER 'IPO1.PARMLIB'
%CRCVER 'IPO1.PROCLIB'
%CRCVER 'TSS.PARMLIB'
%CRCVER 'SYS1.VTAMLST'
%CRCTRAP
/*
```

CRCVER

```
/* REXX */
arg ds rest
if length(ds) = Ø then
   do
     say "Target dataset missing"        /* Send out error message   */
     exit 16                             /* Get out now              */
   end
if length(rest) = Ø then
   do
     x = listdsi(ds)
     if x = Ø then
        do
           rest = sysvolume
           v = "VOL("rest")"
        end
     else
     do
        say "Cannot locate dataset" ds
        exit 16
     end
   end
else v = "VOL("rest")"
say "Building CRCs for" ds
x = outtrap("trap.","*")                 /* Turn on output trapping  */
"CRC32" ds v                             /* Issue CRC command        */
if rc
¬= Ø then
   do
```

© 1999. Reproduction prohibited. Please inform Xephon of any infringement.

```
     say "CRC error, rc=" rc "on" ds rest
     exit 16
   end
x = outtrap("off")                        /* Turn off output trapping */
c = Ø                                     /* Array index              */
d = Ø                                     /* Array index              */
err = Ø

dsnvol  = rest left(strip(ds,'B',"'"),44)
pref = date(s) time()
datex = substr(date(s),3,6)

z1 = '        '

"ALLOC DD(DDNAME) DA('CRC.V"rest"."strip(ds,"B","'") ||,
"(D"datex")') SHR REUSE"
j = Ø
do i = 1 to trap.Ø
   w3 = word(trap.i,3)
   w4 = word(trap.i,4)
   w5 = left(word(trap.i,5),9,' ')
   w6 = left(word(trap.i,6),8,' ')
   w7 = word(trap.i,7)

   select
     when w5 = 'member'    then temp = pref w5 w6 w3 dsnvol
     when w5 = 'directory' then temp = pref w5 z1 w3 dsnvol
     when w5 = 'members'   then temp = pref w5 z1 w3 dsnvol
     when w5 = 'dataset'   then temp = pref w5 z1 w3 dsnvol
     otherwise
        iterate
     end
   j = j + 1
   temp=left(temp,12Ø,' ')                /* Make it 12Ø bytes        */
   line.j = temp
end
say "Writing output"
line.Ø = j
address TSO "EXECIO * DISKW DDNAME (STEM LINE. FINIS)"
"FREE DD(DDNAME)"
```

CRCTRAP

```
/*  REXX */
x = outtrap("trap.","*")                  /* Turn on output trapping  */
"LISTC LEVEL(CRC)"                        /* Issue LISTCAT command    */
if rc ¬= Ø then
   do
     say "LISTCAT error, rc=" rc "on level CRC"
     exit 16
   end
```


© 1999. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (940) 455 7050, fax (940) 455 2492.

```
x = outtrap("off")                          /* Turn off output trapping */
ddn = "MBRLIST"
ww = Ø
do i = 1 to trap.Ø
   w1 = word(trap.i,1)
   w3 = word(trap.i,3)
   w4 = strip(w3)
   if w1 = 'NONVSAM' then do
      x = outtrap("lds.","*")
      "LISTDS '"w3"' MEMBERS"
      x = outtrap("off")
      ok = Ø
      do j = 1 to lds.Ø
         if lds.j = "—MEMBERS—" then do
            ok = j + 1
            leave
            end
      end
      if ok = Ø then do
         say "no members"
         exit 16
         end
      nlast = lds.Ø - 1
      last = lds.Ø
      ww = ww + 1
      comp.ww = w3"("strip(lds.nlast)")"
      ww = ww + 1
      comp.ww = w3"("strip(lds.last)")"
   end
end
comp.Ø = ww
aa = 1
bb = 2

"ALLOC DD(SYSIN) DA('IPO1.PARMLIB(SUPERCRD)') SHR REUSE"
"ALLOC DD(OUTDD) SYSOUT(K) REUSE"
do q = 1 to comp.Ø by 2
      "ALLOC DD(OLDDD) DA('"comp.aa"') SHR REUSE"
      "ALLOC DD(NEWDD) DA('"comp.bb"') SHR REUSE"
      "CALL 'ISP.V4R2MØ.SISPLPA(ISRSUPC)' 'DELTAL'"
      xx = rc
      say "rc="xx "for" comp.aa "vs." comp.bb
      aa = aa + 2
      bb = bb + 2
end
```

© Xephon 1999

© 1999. Reproduction prohibited. Please inform Xephon of any infringement.

Assembler instruction trace

INTRODUCTION

The idea for this instruction trace is based on the ASMTRACE program found on the CBTMODS tape early in the 1980s. Like the original, it is called (BALR 14,15 or BASR 14,15). It uses R14 as the first instruction to be traced. A subsequent call terminates the trace. I started developing the trace in 1990, with refinements being done whenever I required more info, or when new instructions were described in POPS.

This is, however, where all similarity stops. This program is re-entrant, AMODE 31, RMODE ANY. The AMODE/RMODE does not matter if the trace is statically linked to the calling program. However, if a load-and-call sequence is used, and there is a possibility that a mode switch to RMODE 24 can happen, link the trace as RMODE 24. To make things easier, a SETC &RMODE instruction is situated near the top of the source, before the comments.

Trace does a STORAGE OBTAIN for its working storage (LOC=BELOW). If the calling program changes to another protect key (other than zero), an abend 0C4-04 is guaranteed, since Trace now does not have access to its own working storage.

Output is done to DDNAME SYSTRACE. To achieve AMODE/ RMODE independence, access to the DD is done using an ACB. For this reason, SYSTRACE must refer to SYSOUT or an ESDS. If this DD is not present, or DUMMY, then the trace will just return.

I have attempted as far a possible to keep up to date with POPS. String instructions, the relative-and-immediate instructions, MVCLE, etc – are all supported and traced correctly. The cross-memory instructions MVCP and MVCS, as well as access-register code are supported. Ditto for MVCSK and MVCDK.

However, do not attempt to trace VTAM I/O routines. The trace upsets the timing to the extent that a non-zero response is always returned.

© 1999. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (940) 455 7050, fax (940) 455 2492.

© 1999. Reproduction prohibited. Please inform Xephon of any infringement.

```
1PCOMMAND 6.1 LIVE                                          Output                                          98-02-13
15:36:58
 ──────────────────────────────────────────────Lvl 3 Row 1-21 Col 1-123/123
 TCBX05SP JOB  1840  DDname - SYSTRACE  Stepname - PGSTSTB2  Procstep -
 ──────────────────────────────────────────────────────────────────────-
...+....10...+....20...+....30...+....40...+....50...+....60...+....70...+....80...+....90...+....100..+....110..+....120.≥+....130

   CALLED FROM: PGSTSTB2 1998-02-12T15:46:00  Copyright Persetel 1998       TEST PGM 3                          .

       R00=FD000008   R01=00005FF8   R02=00000040   R03=009DE954   R04=009DE930   R05=009E1A20   R06=009BDFF8   R07=FD000000
       R08=009E1C18   R09=809E1458   R10=00000000   R11=00007000   R12=80006000   R13=00008000   R14=80006144   R15=80009308

      AR00=00000000  AR01=00000000  AR02=00000000  AR03=00000000  AR04=00000000  AR05=00000000  AR06=00000000  AR07=00000000
      AR08=00000000  AR09=00000000  AR10=00000000  AR11=00000000  AR12=00000000  AR13=00000000  AR14=00000000  AR15=00000000

       FR0=0000000000000000          FR2=0000000000000000          FR4=0000000000000000          FR6=0000000000000000

  0144:80006144  4100D050      4  LA    R00,0080(,R13)              00008050            (80008050)
  0148:80006148  18E0          4  LR    R14,R00                     00008050             00008050
  014A:8000614A  41000008      4  LA    R01,0008                    00000008            (80000008)
  014E:8000614E  17FF          8  XR    R15,R15                     00000000             00000000
  0150:80006150  A80E0040      2  MVCLE R00,R14,X'0040'             00008058 00000000    00008050 00000000    PAD=X'40'
       OP1 ADDR=00008050 LEN=X'00000008' DATA=4040404040404040
       OP2 ADDR=00008050 LEN=X'00000000' DATA=
  0154:80006154  47F0C1AC      2  B     0428(,R12)                 (800061AC)
  01AC:800061AC  17FF          8  XR    R15,R15                     00000000             00000000
  01AE:800061AE  181D          8  LR    R01,R13                     00008000             00008000
```

Figure 1: Example output from ASMTRACE

OUTPUT

On entry, the trace will attempt to find the entry point (and EP literal) of the calling module. If successful, the offset from the entry point will be displayed. If another subroutine is called, trace will also attempt to find this entry point, and display offsets relative to the start of this routine. An example of the output can be seen in Figure 1, while instructions for output analysis can be seen in Figure 2.

Column	Data	Comments
1-4	Offset from start of CSECT	
6-13	Actual location	Hi-order bit on = AMODE 31
16-28	HEX opcode	
30	Condition code	The values as used for BC instructions, ie 8,4,2,1 or combinations thereof.
32-56	Assembler instruction	Offsets and lengths are decimal
68-133	After-execution operand values	Depends on the instruction.

Figure 2: Output analysis instructions

The output is displayed in the following format:

- Operands are displayed according to the instruction type; for RR-type instructions, the registers involved are displayed. If an even-odd pair is designated, both registers of that pair are displayed.
- For RS or RX instructions, the register(s) are displayed, as well as the effective address. The Effective Field Address (EFA) is always displayed in parentheses, with the high-order bit on if AMODE is 31. If storage from the EFA is referenced, that storage is also displayed. The storage displayed will not necessarily be the full operand length, because the space in the print line is finite.
- For SS instructions, the EFA and storage for both operands are displayed. However, for MVC, only the first operand EFA and storage, and the second operand EFA are displayed. For OC and XC, a check is made if first and second operands are the same, and if so, more data from the area can be displayed.

© 1999. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (940) 455 7050, fax (940) 455 2492.

- Certain instructions operate on multiple registers, like LM, STM, BAKR, PR, LAM and STAM. For these instructions, ALL registers and/or access registers are displayed. This is also done for CSD (Compare and Swap Double), since storage as well as a large number of registers are referenced.
- For MVCL, CLCL, MVCLE, and CLCLE, at least the first 88 bytes of each operand will be displayed.
- For SVC and PC instruction, registers and ARs will be dumped before and after the call. As much info as possible is given as regards the macro that generated the SVC or PC call. An attempt is also made to relate a macro to the SVC or PC call where possible, as well as any extra info. For example, on OPEN/CLOSE, the DDNAME and MODE values are displayed.
- SVC 3 (EXIT), 7(XCTL), and 55(EOV) cause trace to lose control. For this reason, if it detects any of these, it will react as if a termination call had been made from the calling program: clean up, then restore all registers and branch to the SVC instruction address.

OPERATIONAL ENVIRONMENT

No guarantees are given that this trace will work under all conditions. Certain instructions will result in a pseudo-0C1 abend. These instructions should not occur in normal situations, so should not be a great worry. Some of these are LPSW, PALB, and PTLB. I have just not had a situation where I could conveniently test these instructions without possibly dropping the entire machine, hence S0C1. This trace will operate only on MVS/ESA (3.1 or higher), and OS/390.

MACROS

I use a lot of macros to make the code more readable. Most of these are included in the source code. The exceptions are the macros in copybooks PPFC14M0 and PPFGBLC0 (the latter is used by the first). These are the CONCEPT-14 macros found on old share tapes, with extensions. These macros include the following structures:

© 1999. Reproduction prohibited. Please inform Xephon of any infringement.

```
IF-ELSE-ENDIF
DO-DOEXIT-ENDDO
STRTSRCH-EXITIF-ORELSE-ENDLOOP-ENDSRCH
SELECT-WHEN-ENDSEL
```

The usage of these macros is described in the *High-Level Assembler Toolkit User Guide: Using Structured Programming Macros*. Note, that this set does *not* allow lower case condition codes. Also, the THEN statement is not used in an IF construct. It was also adapted to correctly process the IAC test. My version of the SELECT structure is totally different, it conforms more to the REXX version:

```
SELECT {EVERY}
WHEN condition 1
     Process 1
WHEN condition 2
     Process 2
     (
     (
     (
WHEN condition N
     Process N
WHEN NONE
Process when none of the preceding is true.
ENDSEL
```

This structure was primarily developed to bypass the problem of multiple indents with nested IF-ELSE-ENDIF structures. The conditions thus apply exactly as described for the IF structures. If the EVERY keyword is supplied, the structure generates code similar to a series of consecutive IF-ENDIF structures, ie every WHEN condition is tested, and each condition may be influenced by any preceding process. NONE cannot be used if the EVERY keyword is used.

PROGRAM LOGIC

The program logic is shown below:

1 Save caller registers.

2 Obtain working storage below, since caller may switch to 24-bit AMODE. Included in this is a register table, a new save area, as well a table for access registers.

3 Copy the register values from caller's save area, save R13 in the appropriate slot, point R13 at the new save area, and save the access registers.

 © 1999. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (940) 455 7050, fax (940) 455 2492.

4 Use R14 from caller as the first instruction to be traced.

5 Loop until the next instruction address is the same as trace's entry point. This signals end of trace.

6 Most instructions will be EXed, except where the instruction could generate a branch, in which case the operation is emulated. Registers used in the actual execution will be a combination of R2-R5. Certain instructions (SVC, the xSCH instructions, PC, PR, etc) require the use of specific registers. This is catered for.

7 After execution of the instruction, get the condition code using IPM. If this instruction could have changed the condition code, store it somewhere. Save any register changes in the appropriate slot in the register table.

8 Display offset (not necessarily), instruction address, opcode, condition code, Assembler instruction, and after-execution operands.

9 Increment the instruction pointer by the appropriate number of bytes.

10 Cleanup – close SYSTRACE DCB, move the current register table to caller save area, point R13 at that save area, release storage and return.

ASMTRACE

```
         PRINT OFF
&RMODE   SETC  '24'                          .CHANGE TO 'ANY' IF NEEDED.
*LKDPARM=RENT
*STDUSE=NO
*PRTUSE=NO
         COPY  PPFC14MØ
         PRINT ON,NOPRINT NOGEN
         PUNCH ' ALIAS ASMTRACE'
*********************************************************************
* NOTE: H-ASSEMBLER REQUIRED FOR THIS ASSEMBLY.                     *
*********************************************************************
*NOTE:   THIS PROGRAM WILL WORK ONLY ON MVS/ESA OR OS/39Ø           *
*                                                                   *
*        THIS ROUTINE IS CALLED AS A SUBROUTINE, AND WILL START     *
*        TRACING INSTRUCTIONS FROM THE FIRST INSTRUCTION AFTER      *
*        THE CALL. THE CALL MUST A VIA BALR OR BASR                 *
*                                                                   *
```

© 1999. Reproduction prohibited. Please inform Xephon of any infringement.

```
*        A SUBSEQUENT CALL TO ASMTRACE WILL TURN THE TRACE OFF.       *
*        EXCEPTIONS: THE USE OF THE FOLLOWING SVCS WILL ALSO         *
*                    TURN TRACING OFF:                                *
*                         3 (EXIT)                                    *
*                         7 (XCTL)                                    *
*                         55 (EOV).                                   *
*                                                                     *
*        IF NO 'SYSTRACE' DD IS FOUND, ASMTRACE WILL JUST             *
*        RETURN WITHOUT TAKING ACTION. PLEASE NOTE THAT THE           *
*        'SYSTRACE' DD MUST REFER TO SYSOUT, SINCE THE TRACE          *
*        RECORDS ARE WRITTEN USING AN ACB. ALTERNATIVELY, THE         *
*        SYSTRACE DD CAN REFER TO AN ESDS WITH LRECL=133. FOR         *
*        BROWSING THE ESDS, REFER TO PROGRAM PIFBROIØ.                *
*                                                                     *
*        THIS TRACE IS FULLY RE-ENTRANT, SO LINK AS RENT.             *
*                                                                     *
*        THE AMODE=31,RMODE=ANY, THUS MAKING THE SUBROUTINE           *
*        COMPELETELY INDEPENDENT OF THE CALLER'S AMODE/RMODE,         *
*        IF ASMTRACE IS STATICALLY LINKED TO THE CALLING PROGRAM.     *
*                                                                     *
*        IF A LOAD-AND-BASR APPROACH IS USED, IT IS REQUIRED THAT     *
*        THE MODULE BE LINKED WITH RMODE=24 IF THE CALLING MODULE     *
*        (OR SUBROUTINES) COULD AT ANY STAGE SWITCH TO 24-BIT AMODE.  *
*                                                                     *
*        WORKING STORAGE WILL BE OBTAINED BELOW THE LINE.             *
*                                                                     *
*        MOST INSTRUCTIONS WILL BE TRACEABLE; HOWEVER, NO GUARANTEE   *
*        IS GIVEN THAT ALL INSTRUCTIONS WILL BE TRACED CORRECTLY.     *
*        THIS APPLIES ESPECIALLY TO THE SUBCHANNEL COMMANDS (SSCH,    *
*        MSCH, ETC). THE CODE EXISTS TO TRACE THEM, BUT HAS NEVER     *
*        BEEN TESTED.                                                 *
*                                                                     *
*        ONE INSTRUCTION IS GUARANTEED TO GIVE PROBLEMS: SPKA WITH    *
*        KEY <> Ø AND <> CALLER'S ORIGINAL KEY. THE REASON IS THAT,   *
*        IF A KEY IS CHANGED FROM THE CALLER'S CURRENT KEY, ASMTRACE  *
*        LOSES ACCESS TO ITS OWN WORKING STORAGE, RESULTING IN A      *
*        SØC4-Ø4 ABEND. THE SAME APPLIES TO MODESET WITH A NON-ZERO   *
*        KEY OTHER THAN THE CALLER'S.                                 *
*                                                                     *
*        TRCACING OF THE FOLLOWING INSTRUCTIONS WILL RESULT IN A      *
*        SØC1 ABEND: LPSW, PTLB, PALB, TRACE, STURA, LURA, TAR        *
*                                                                     *
*        IF THE CPU SUPPORTS IMMEDIATE-AND-RELATIVE FEATURE, THESE    *
*        INSTUCTIONS WILL BE TRACE CORRECTLY (LHI, CHI, MHI, AHI,     *
*        BRC, BRXH, BRXLE, BRCT)                                      *
*                                                                     *
*        PRE-ESA CROSS-MEMORY INSTRUCTIONS WILL BE TRACED CORRECTLY.  *
*                                                                     *
*        PC INSTRUCTIONS AND SVC INSTRUCTIONS WILL BE EXECUTED AND    *
*        THE RESULTS SHOWN, BUT THE CODE OF THE RELATED ROUTINES      *
*        WILL NOT BE TRACED. REGISTERS BEFORE AND AFTER THE CALL      *
*        WILL BE DISPLAYED.                                           *
*                                                                     *
```


© 1999. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (940) 455 7050, fax (940) 455 2492.

```
*        AR CODE IS SUPPORTED FOR MOST INSTRUCTIONS. HOWEVER, NO      *
*        GUARANTEE IS GIVEN THAT THIS APPLIES TO ALL INSTRUCTIONS.    *
*********************************************************************
         EJECT
*********************************************************************
*        REGISTERS UPON ENTRY:                                        *
*                                                                     *
*        R15 = ENTRY POINT OF ASMTRACE                                *
*        R14 = RETURN ADDRESS                                         *
*        R13 = ADDRESS OF 72-BYTE SAVE AREA                           *
*                                                                     *
*        NOTE THAT ASMTRACE MUST BE CALLED, NOT LINKED. IF IT         *
*        IS LINKED,  R14 WILL POINT TO AN EXIT SVC, WHICH IS          *
*        NO BIG HELP                                                  *
*                                                                     *
*        TRACE WILL ATTEMPT TO FIND THE ENTRY POINT LITERAL OF        *
*        THE CALLER.  IF THE CALLER DOES NOT USE STANDARD             *
*        LINKAGE CONVENTION, OR USES MULTIPLE SAVE AREAS,             *
*        IT WILL NOT BE POSSIBLE TO DO THIS.                          *
*                                                                     *
*        IF THE ENTRY POINT LITERAL HAS BEEN FOUND, THE OFFSET        *
*        FROM THE START OF THE CSECT WILL BE SHOWN IN THE TRACE,      *
*        AS WELL AS THE INSTRUCTION COUNTER.                          *
*                                                                     *
*        DSECTS/MAPS:                                                 *
*        IHAPSA, IKJTCB, IEFTIOT1, REGEQU                             *
*                                                                     *
*        OTHER MACROS:                                                *
*        GENCB, MODCB, OPEN, CLOSE, PUT                               *
*                                                                     *
*        INLINE MACROS:                                               *
*        ALL 'CONCEPT-14' MACROS (IF-ELSE-ENDIF, DO-DOEXIT-ENDDO,     *
*                         STRTSRCH-EXITIF-ORELSE-ENDLOOP-ENDSRCH)     *
*                         COPYBOOK PPFC14MØ.                          *
*        EXTENSION OF CONCEPT-14 MACROS (SELECT-WHEN-ENDSEL), ALSO    *
*        IN  COPYBOOK PPFC14MØ.                                       *
*        PERFORM STRUCTURES                                           *
*                (PERF, MODENTRY, MODEXIT, RETSTACK,                  *
*                 INIT_RETURN_STACK)                                  *
*        TST31    (SETS HI-ORDER BIT OF DESIGNATED REG IF 31-BIT)     *
*        RUN_INST (EX INSTRUCTION, SAVE COND CODE)                    *
*        SHOW_EFA (DISPLAY EFFECTIVE FIELD ADDRESS)                   *
*********************************************************************
         EJECT
         MACRO
         RUN_INST
         EX    Ø,CODEFLD
         IPM   R15                        .GET CC BITS & PGM MASKS
         IF    TM,FLAGS,CCBIT,O           .CAN INSTR CHANGE THE CC?
            STCM  R15,B'1ØØØ',REALCC      .YES - STORE IT
         ENDIF
```

© 1999. Reproduction prohibited. Please inform Xephon of any infringement.

```
         MEND
         EJECT
         MACRO
         TST31 &R,&LIST=NO
.*-------------------------------------------------------------------*.
.* THIS MACRO WILL DO A BSM 'R',Ø WHICH WILL SET THE HI-ORDER BIT    *.
.* OF 'R' IF WE ARE CURRENTLY IN 31-BIT MODE.                        *.
.* HOWEVER, BSM DOES NOT WORK FOR RØ, SO WE SWAP RØ & R15, DO THE    *.
.* BSM WITH R15, AND SWAP RØ & R15 BACK.                             *.
.*-------------------------------------------------------------------*.
         PUSH  PRINT
         AIF   ('&LIST'(1,1) EQ 'Y').PRINTON
         PRINT OFF
         AGO   .GENCODE
.PRINTON ANOP
         PRINT ON,GEN
.GENCODE ANOP
         AIF   ('&R' NE 'Ø').NOTRØ     .NOT RØ, STANDARD BSM
         XR    15,Ø                    .SWAP...
         XR    Ø,15                    . RØ...
         XR    15,Ø                    .  AND R15
         BSM   15,Ø                    .SET R15'S HI-ORDER BIT
         XR    15,Ø                    .AND SWAP..
         XR    Ø,15                    . THEM..
         XR    15,Ø                    .  BACK
         POP   PRINT
         MEXIT
.NOTRØ   ANOP
         BSM   &R,Ø
         POP   PRINT
         MEND
         EJECT
         MACRO
&LBL     PERF  &MOD
.*-------------------------------------------------------------------*.
.* THIS MACRO IS USED TO CALL A SUB-PROCEDURE.                       *.
.* SINCE THIS PROGRAM IS SO BIG, A SIMPLE BAS R14,&MOD WILL          *.
.* NOT ALWAYS WORK, HENCE THE BASR CONSTRUCT                         *.
.*-------------------------------------------------------------------*.
         L     R15,=A(&MOD)
&LBL     BASR  R14,R15
         MEND
         SPACE 3
         MACRO
&LBL     MODENTRY  &NEWBASE=,&LIST=NO,&BAKR=NO,&INITBASE=
.*-------------------------------------------------------------------*.
.* THIS MACRO IS THE ENTRY TO A SUB-PROCEDURE.                       *.
.* PARAMETERS:                                                       *.
.*       BAKR:     IF NO, USE RETURN STACK TO SAVE GPR 14, ELSE USE  *.
.*                 BAKR                                            *.
.*       NEWBASE:  IF NON-BLANK, WILL PRIMT NEWBASE FROM GPR 15,     *.
.*                 THEN ISSUE 'USING &LBL,&NEWBASE'                  *.
```


© 1999. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (940) 455 7050, fax (940) 455 2492.

```
.*       INITBASE: MUTUALLY EXCLUSIVE WITH NEWBASE. IF NON-BLANK,    *.
.*                 THE FOLLOWING CODE IS GENERATED:                  *.
.*                     BASR  &INITBASE,Ø                             *.
.*                     USING *,&INITBASE                             *.
.*                     L     &INITBASE,=A(&LBL)                      *.
.*                     USING &LBL,&INITBASE                          *.
.*       LIST:     IF YES, ISSUE 'PRINT ON,GEN', ELSE 'PRINT OFF'    *.
.*                                                                   *.
.* THIS MACRO CANNOT BE NESTED - IE YOU CANNOT HAVE TWO MODENTRY     *.
.* STATEMENTS WITHOUT AN INTERVENING MODEXIT - THIS IS ENFORCED.     *.
.*-------------------------------------------------------------------*.
         GBLC  &CURMOD
         GBLB  &BAKROFF
         AIF   ('&NEWBASE.&INITBASE' EQ '').NOLTORG
         PUSH  PRINT
         AIF   ('&LIST' EQ 'YES').PRTALL
         PRINT OFF
.PRTALL  ANOP
         LTORG
         POP   PRINT
.NOLTORG ANOP
         AIF   ('&CURMOD' NE '').BUSY
&CURMOD  SETC  '&LBL'
         DS    ØH
         DC    C'  &LBL  '
&LBL     DS    ØH
&BAKROFF SETB  ('&BAKR'(1,1) NE 'Y')
         AIF   (&BAKROFF).NEWBASE
         BAKR  R14,Ø
.NEWBASE ANOP
         AIF   ('&NEWBASE' EQ '').INITBASE
         DROP  &NEWBASE
         LR    &NEWBASE,R15
         USING &LBL,&NEWBASE
         AGO   .NOBASE
.INITBASE ANOP
         AIF   ('&INITBASE' EQ '').NOBASE
         DROP  &INITBASE
         BASR  &INITBASE,Ø
         USING *,&INITBASE
         L     &INITBASE,=A(&LBL)
         USING &LBL,&INITBASE
         AIF   (&BAKROFF).NOBASE
         MEXIT
.NOBASE  ANOP
         L     R15,@NXTRET@
         ST    R14,Ø(,R15)
         LA    R15,4(,R15)
         ST    R15,@NXTRET@
         MEXIT
         MEXIT
.BUSY    ANOP
```

© 1999. Reproduction prohibited. Please inform Xephon of any infringement.

```
         MNOTE 8,'''MODEXIT'' REQUIRED TO CLOSE MODULE ''&CURMOD'''
         MEND
         SPACE 3
         MACRO
&LBL     MODEXIT
.*-------------------------------------------------------------------*.
.* THIS MACRO IS THE EXIT FROM A SUB-PROCEDURE.                      *.
.* IF THE PRECEDING MODENTRY HAD BEEN CALLED WITH 'BAKR=YES', THEN   *.
.* A 'PR' INSTRUCTION WILL BE GENERATED,  ELSE THE RETURN STACK IS   *.
.* USED.                                                             *.
.*-------------------------------------------------------------------*.
         GBLC  &CURMOD
         GBLB  &BAKROFF
         AIF   ('&CURMOD' EQ '').NOTBUSY
&CURMOD  SETC  ''
&LBL     DS    ØH
         AIF   (&BAKROFF).RETURN
         PR
         MEXIT
.RETURN  ANOP
         L     R15,@NXTRET@
         S     R15,=F'4'
         ST    R15,@NXTRET@
         L     R14,Ø(,R15)
         BR    R14
         MEXIT
.NOTBUSY ANOP
         MNOTE 8,'''MODEXIT'' NOT PRECEDED BY ''MODENTRY'''
         MEND
         EJECT
         MACRO
&LBL     INIT_RETURN_STACK
.*-------------------------------------------------------------------*.
.* THIS MACRO INITIALIZES THE RETURN STACK FOR USE BY MODENTRY AND   *.
.* MODEXIT.                                                          *.
.*-------------------------------------------------------------------*.
&LBL     LA    R15,RETSTACK
         ST    R15,@NXTRET@
         S     R15,=F'16'
         MVC   Ø(16,15),=CL16'RETURN STACK'
         MEND
         SPACE 3
         MACRO
         RETSTACK  &SIZE=64
.*-------------------------------------------------------------------*.
.* THIS MACRO DEFINES THE RETURN STACK                               *.
.* PARAMETER SIZE = SUMBER OF FULLWORDS TO DEFINE FOR THE STACK.     *.
.*-------------------------------------------------------------------*.
@NXTRET@ DS    F
         DS    2D
RETSTACK DS    &SIZE.A
         MEND
```


© 1999. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (940) 455 7050, fax (940) 455 2492.

```
         SPACE 3
         MACRO
         SHOW_EFA &FROM=,&TO=,&FOR=,&MAX=
.*-------------------------------------------------------------------*.
.* THIS MACRO SETS UP CERTAIN REGISTERS, THEN CALLS PROCEDURE       *.
.* SHOW_EFA                                                         *.
.*       PRAMETERS:                                                 *.
.*             FROM: ADDRESS FROM WHICH TO DISPLAY                  *.
.*             TO:   ADDRESS OF THE OUTPUT AREA                     *.
.*             FOR:  LENGTH OF DATA TO DISPLAY. IF ZERO, SHOW ONLY  *.
.*                   THE ADDRESS.                                   *.
.*             MAX:  DO NOT SHOW MORE THAN THIS VALUE,  EVEN IF THE *.
.*                   LENGTH IS MORE.                                *.
.*-------------------------------------------------------------------*.
         AIF   ('&FROM' EQ '').NOFROM
         LA    R8,&FROM
.NOFROM  AIF   ('&FOR' EQ '').NOFOR
         LA    R5,&FOR
.NOFOR   AIF   ('&TO' EQ '').NOTO
         LA    R6,&TO
.NOTO    AIF   ('&MAX' EQ '').NOMAX
         IF    C,R5,GT,=AL4(&MAX)
            LA    R5,&MAX
         ENDIF
.NOMAX   PERF  SHOW_EFA
         MEND
         SPACE 3
         MACRO
         SHOW_AR &NEW,&FROM=,&TO=
         AIF   ('&FROM' EQ '').NOFROM
         LA    R8,&FROM
.NOFROM  AIF   ('&TO' EQ '').NOTO
         LA    R6,&TO
.NOTO    ANOP
         AIF   ('&NEW' EQ 'NEW').NEWARS
         LA    R5,AR_OLD
         AGO   .GOSHOW
.NEWARS  ANOP
         LA    R5,AR_SAVE
.GOSHOW  ANOP
         PERF  SHOW_AR
         MEND
         EJECT
         GBLC  &SYSSPLV
         SPLEVEL TEST
         AIF   ('&SYSSPLV' GE '3').SPOK
         MNOTE 8,'ASMTRACE WILL ONLY WORK ON MVS ESA'
.SPOK    ANOP
ASMTRACE CSECT
ASMTRACE AMODE 31
ASMTRACE RMODE &RMODE
         SAVE  (14,12),,ASMTRACE..DATE=&SYSDATC..TIME=&SYSTIME..SP&SYSS+
```

© 1999. Reproduction prohibited. Please inform Xephon of any infringement.

```
            PLV..INSTRUCTION-TRACE-BY-PIETER-WIID-PERSETEL-PRETORIA
         LA    R12,Ø(,R15)             .CLEAR ALL UNWANTED BITS
         USING ASMTRACE,R12,R11
         LA    R11,2Ø48(,R12)
         LA    R11,2Ø48(,R11)
         IPM   R8                     .GET CURRENT CC
         EPAR  R2
         ESAR  R3
         XR    R4,R4
         SELECT EVERY
         WHEN  IAC,R4,NZ                    .SECONDARY/HOME/AR ?
            SAC   Ø                         .SET TO PRIMARY MODE
         WHEN  CR,R2,NE,R3
            SSAR  R2                        .SEONDARY=PRIMARY
         ENDSEL
         STORAGE OBTAIN,LENGTH=WSLEN,LOC=BELOW,COND=NO,BNDRY=PAGE
         LR    R7,R1
         USING WRKSTOR,R7
         LR    R1,R13                  .BACKUP TRACEE'S SAVE AREA PTR
         LR    R13,R7                  .POINT R13 AT MY SA
         MVC   REGTBL(13*4),2Ø(R1)     .MOVE RØ-R12 TO WRK
         ST    R1,REGTBL+13*4          .AND R13
         MVC   REGTBL+14*4(8),12(R1)   .AND R14-R15
         STAM  RØ,R15,AR_SAVE
         LAM   RØ,R15,=16F'Ø'
         SELECT EVERY
         WHEN  CR,R2,NE,R3                  .PRIMARY A/S NE SEONDARY
            SSAR  R3                        .SET SECONDARY AS ON ENTRY
         WHEN  LTR,R4,R4,NZ                 .NOT PRIMARY MODE?
            SAC   Ø(R4)                     .THEN SET IT
         ENDSEL
         INIT_RETURN_STACK             .INIT PERFORM/RET STACK
         STCM  R8,B'1ØØØ',REALCC       .SAVE CC & SYSTEM MASKS
         PERF  INIT                    .INIT CONSTANTS LIKE DCB'S
         DO    WHILE=(CR,R9,NE,R12)    .AM I BEING CALLED AGAIN?
            PERF  TRACE_IT             .NO,SO TRACE THE INSTRUCTION
         ENDDO
BREAK_LOOP     DS  ØH
         PERF  CLEANUP                 .MOSTLY, CLOSE THE SYSPRINT DCB
         EJECT
RETURN   DS    ØH
         L     R14,REGTBL+14*4
         IF    CLC,=X'Ø1Ø1',EQ,Ø(R14)
            LM    R2,R3,REGTBL
            L     R4,REGTBL+15*4
            STORAGE RELEASE,LENGTH=WSLEN,ADDR=(7)      .FREE WRKSTOR
            PR
         ENDIF
         L     R13,REGTBL+13*4         .GET TRACEE'S R13
         MVC   2Ø(13*4,R13),REGTBL     .MOVE REGS Ø-12 TO SAVE AREA
         MVC   12(8,R13),REGTBL+14*4   .AS WELL AS R14,R15
         LAM   RØ,R15,AR_SAVE
```


© 1999. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (940) 455 7050, fax (940) 455 2492.

```
         EPAR  R2
         ESAR  R3
         XR    R4,R4
         SELECT EVERY
         WHEN  IAC,R4,NZ
            SAC   Ø
         WHEN  CR,R2,NE,R3
            SSAR  R2
         ENDSEL
         STORAGE RELEASE,LENGTH=WSLEN,ADDR=(7)      .FREE WRKSTOR
         SELECT EVERY
         WHEN  CR,R2,NE,R3
            SSAR  R3
         WHEN  LTR,R4,R4,NZ
            SAC   Ø(R4)
         ENDSEL
         RETURN (14,12),RC=Ø
         EJECT
INIT     MODENTRY
         PERF  KILLXMS
         LA    R1,CUREP
         LA    R1,EPSTACK-CUREP(,R1)
         ST    R1,EPSTACK@
         STM   R2,R4,XMSSTAT
         GENCB BLK=ACB,DDNAME=SYSTRACE,MACRF=(ADR,OUT),MF=(G,CALLPARM),+
               RMODE31=ALL,WAREA=(S,ACB),LENGTH=ACB_SIZE
         GENCB BLK=RPL,ACB=(S,ACB),AREA=(S,PRTLINE),AREALEN=133,       +
               MF=(G,CALLPARM),RECLEN=133,OPTCD=(ADR,MVE),             +
               WAREA=(S,RPL),LENGTH=RPL_SIZE
         LA    RØ,OPENLST              .SYSTRACE DCB, START OF CONSTS
         L     R14,=A(MODELS)          .MODEL CONSTANS
         L     R1,=A(MODELSZ)          .LENGTH OF MODEL CONSTANTS
         LR    R15,R1                  .SRC LEN=DEST LEN
         MVCL  RØ,R14                  .AND MOVE
         DEVTYPE =CL8'SYSTRACE',DUB
         LTR   R15,R15                      .DDN 'SYSTRACE' NOT FOUND?
         BNZ   RETURN
         OC    DUB,DUB                      .DD DUMMY?
         BZ    RETURN
         OPEN  ACB,MF=(E,OPENLST),MODE=31
         IF    C,R15,GT,=F'4'
            SHOWCB ACB=(S,ACB),AREA=(S,DUB),LENGTH=4,FIELDS=ERROR,     +
               MF=(G,CALLPARM)
            L     R3,DUB
            WTO   'TRACE ERROR: CAN''T OPEN DDNAME SYSTRACE'
            ABEND 111,DUMP
         ENDIF
         LM    R2,R4,XMSSTAT
         PERF  RSETXMS
         L     R1,REGTBL+14*4          .CALLER'S RET. ADDR = ADDR AT
         LA    R9,Ø(,R1)               .WHICH TO START TRACING
         ST    R9,NEW_IPTR             .SAVE THE INST PTR FOR DISASM
```

© 1999. Reproduction prohibited. Please inform Xephon of any infringement.

```
         STD   RØ,FLTRØ                .SAVE..
         STD   R2,FLTR2                .ALL THE...
         STD   R4,FLTR4                .FLOATING POINT..
         STD   R6,FLTR6                .REGISTERS
         MVI   PRTLINE,X'4Ø'           .CLEAR THE PRINT LINE
         MVC   PRTLINE+1(L'PRTLINE-1),PRTLINE
         L     R1,REGTBL+13*4          .GET CALLER'S R13
         IF    CLC,=C'F1SA',EQ,4(R1)
            EREG  R14,R15     .GET ENTRY REGS 14&15 FROM LINKAGE STK
            LR    R4,R14
            LR    R5,R15
         ELSE
            L     R1,4(,R1)               .AND GO 1 UP THE CHAIN
            LM    R4,R5,12(R1)            .GET HIS RET ADDR + EPA
         ENDIF
         PERF  SHOW_EP             .SHOW EP LITERAL,SAVE EPA& RET ADDR.
         PERF  DUMPREGS                .SHOW ALL GENERAL REGS
         PERF  DUMP_ARS
         PERF  DUMP_FLT                .AND FLOATING PT REGS
         MODEXIT
         EJECT
TRACE_IT MODENTRY
         MVI   PRTLINE,X'4Ø'
         MVC   PRTLINE+1(L'PRTLINE-1),PRTLINE
         MVC   EXD_LINE,PRTLINE
         MVC   AR_LINE,PRTLINE
         MVC   OLDREGS(16*4),REGTBL    .BACKUP REGS
         MVC   AR_OLD(16*4),AR_SAVE
         XR    R1,R1
         IC    R1,Ø(,R9)               .GET OPPCODE
         SLL   R1,1                    .* 2
         L     R15,=A(OPFLAGS)
         LH    RØ,Ø(R1,R15)            .GET FLAGS FOR THIS OP
         STH   RØ,FLAGS
         SELECT
         WHEN  TM,FLAGS+1,RRBIT,O      .RR-INSTUCTION?
            MVC   XCELL(2),Ø(R9)
         WHEN  TM,FLAGS+1,SSBIT,O      .SS-INSTUCTION?
            MVC   XCELL(6),Ø(R9)
         WHEN  NONE                    .THEN IT MUST BE RS, RX OR SI
            MVC   XCELL(4),Ø(R9)       .OR EXTENDED OPCODE
         ENDSEL
         IF    CLI,XCELL,EQ,X'44'      .EX INST?
            PERF  EXEC_EX              .YES, SPECIAL CASE
         ELSE
            MVC   CODEFLD,XCELL        .MOVE TO WRK AREA
            PERF  SWITCHER             .TRACE DISPATCHER
            IF    CLC,AR_LINE,NE,=CL133' '
               XC    AR_LINE,PRTLINE
               XC    PRTLINE,AR_LINE
               XC    AR_LINE,PRTLINE
               PERF  WRITE                .WRITE TRACE LINE
```


© 1999. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (940) 455 7050, fax (940) 455 2492.

```
            MVC   PRTLINE,AR_LINE
         ENDIF
         PERF  WRITE                .WRITE TRACE LINE
         SELECT EVERY
         WHEN  TM,FLAGS+1,LMSTMBIT,O, .LM,STM, OR LOTS OF REGS?     +
            OR,CLC,=X'Ø1Ø1',EQ,XCELL .PR?
            PERF  DUMPREGS
         WHEN  TM,FLAGS+1,ARBIT,O,OR,    .ACCESS REGISTERS?         +
            CLC,=X'Ø1Ø1',EQ,XCELL        .PR?
            IF    TM,FLAGS+1,LMSTMBIT,Z
               PERF  WRITE
            ENDIF
            PERF  DUMP_ARS
         ENDSEL
      ENDIF
      SELECT
      WHEN  CLC,=X'Ø5EF',EQ,XCELL,OR,    .BALR 14,15?               +
            CLC,=X'ØDEF',EQ,XCELL,OR,    .BASR 14,15?               +
            CLC,=X'ØCEF',EQ,XCELL        .BASSM 14,15
         LM    R4,R5,REGTBL+14*4    .GET R14 & R15
         PERF  SHOW_EP              .SHOW EP LIT, SAVE EPA&RET ADDR
      WHEN  CLC,=X'Ø7FE',EQ,XCELL     .BR14?
         L     R4,OLDREGS+14*4       .GET R14
         PERF  SHOW_RET             .SEE IF RETURN FROM CURRENT EP
      WHEN  CLI,XCELL,EQ,X'ØB'      .BSM?
         IC    RØ,XCELL+1
         N     RØ,=F'15'
         IF    C,RØ,EQ,=F'14'       .BSM X,14
            L     R4,OLDREGS+14*4    .GET R14
            PERF  SHOW_RET          .SEE IF RETURN FROM CURRENT EP
         ENDIF
      WHEN  CLC,=X'Ø1Ø1',EQ,XCELL       .PR?
         LR    R4,R9
         PERF  SHOW_RET             .SEE IF RETURN FROM CURRENT EP
      ENDSEL
      LA    R9,Ø(,R9)              .ZERO HI-ORDER BITS
      ST    R9,NEW_IPTR
      MODEXIT
      EJECT
SHOW_EP  MODENTRY
      STM   R4,R5,XMSSTAT
      IF    CLI,Ø(R5),EQ,X'47',AND,TM,1(R5),X'FØ',O  .B ROUND LIT??
         PERF  WRITE
         LM    R4,R5,XMSSTAT
         IF    ICM,R3,15,CUREP,Z
            L     R3,EPSTACK@          .LOOKS LIKE IT,
            MVC   PRTLINE(14),=C' CALLED FROM: '
         ELSE
            LA    R3,L'EPSTACK(,R3)
            LA    R15,L'EPSTACK
            MH    R15,=H'4Ø'
            A     R15,EPSTACK@
```

© 1999. Reproduction prohibited. Please inform Xephon of any infringement.

```
            IF    CR,R3,GT,R15
               WTO   'EP LITERAL STACK OVERFLOW'
               ABEND 444,DUMP
            ENDIF
            MVC   PRTLINE(14),=C'  EP LITERAL: '
         ENDIF
         ST    R3,CUREP                .SO PRIME THE EP STACK
         MVC   Ø(L'EPSTACK,R3),=CL133' '
         LA    R4,Ø(,R4)
         LA    R5,Ø(,R5)
         STM   R4,R5,Ø(R3)
         XR    R1,R1
         IC    R1,4(,R5)
         IF    C,R1,GT,=A(L'EPSTACK-8)
            LA    R1,L'EPSTACK-8
         ENDIF
         BCTR  R1,Ø
         EX    R1,MOVE_LIT
         MVC   PRTLINE+14(1Ø2),8(R3)
         PERF  WRITE
      ENDIF
      MODEXIT
      EJECT
SHOW_RET MODENTRY
      IF    ICM,R3,15,CUREP,NZ
         LA    R4,Ø(,R4)
         IF    C,R4,EQ,Ø(,R3)
            PERF  WRITE
            L     R3,CUREP
            MVC   PRTLINE(14),=C' RETURN FROM: '
            MVC   PRTLINE+14(36),8(R3)
            PERF  WRITE
            MVC   PRTLINE(133),=133C'-'
            PERF  WRITE
            L     R3,CUREP
            S     R3,=AL4(L'EPSTACK)
            LA    R15,L'EPSTACK
            MH    R15,=H'35'
            L     R14,EPSTACK@
            SR    R14,R15
            IF    CR,R3,LT,R14
               WTO   'EP LITERAL STACK UNDERFLOW'
               ABEND 555,DUMP
            ENDIF
            ST    R3,CUREP
            L     R14,EPSTACK@
            IF    CR,R3,LT,R14
               L     R1,REGTBL+13*4
               IF    ICM,R1,15,4(R1),NZ
                  PERF  DADS_EP
               ENDIF
            ELSE
```


© 1999. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (940) 455 7050, fax (940) 455 2492.

```
                  MVC   PRTLINE(21),=C' CURRENT EP LITERAL: '
                  MVC   PRTLINE+21(L'EPSTACK-8),8(R3)
                  PERF  WRITE
                  PERF  WRITE
               ENDIF
            ENDIF
         ENDIF
         MODEXIT
         EJECT
DADS_EP  MODENTRY
         LM    R4,R5,12(R1)
         IF    CLI,Ø(R5),EQ,X'47',AND,TM,1(R5),X'FØ',O
            MVC   Ø(L'EPSTACK,R3),=CL8Ø' '
            LA    R4,Ø(,R4)
            LA    R5,Ø(,R5)
            STM   R4,R5,Ø(R3)
            XR    R1,R1
            IC    R1,4(,R5)
            IF    C,R1,GT,=A(L'EPSTACK-8)
               LA    R1,L'EPSTACK-8
            ENDIF
            BCTR  R1,Ø
            EX    R1,MOVE_LIT
            MVC   PRTLINE(21),=C' CURRENT EP LITERAL: '
            MVC   PRTLINE+21(L'EPSTACK-8),8(R3)
            PERF  WRITE
            PERF  WRITE
         ENDIF
         MODEXIT
         EJECT
WRITE    MODENTRY
         PERF  KILLXMS
         PUT   RPL=RPL
         IF    LTR,R15,R15,NZ
            WTO   'ASMTRACE: SYSTRACE PUT ERROR; R14=RPL FDBK CODE'
            SHOWCB RPL=(S,RPL),AREA=(S,DUB),LENGTH=4,FIELDS=FDBK,      +
               MF=(G,CALLPARM)
            L     R14,DUB
            DC    H'Ø'
         ENDIF
         MVI   PRTLINE,X'4Ø'
         MVC   PRTLINE+1(L'PRTLINE-1),PRTLINE
         PERF  RSETXMS
         MODEXIT
         EJECT
KILLXMS  MODENTRY
         EPAR  R2
         ESAR  R3
         XR    R4,R4
         SELECT EVERY
         WHEN  IAC,R4,NZ
            SAC   Ø
```

© 1999. Reproduction prohibited. Please inform Xephon of any infringement.

```
         WHEN  CR,R2,NE,R3
            SSAR  R2
         ENDSEL
         MODEXIT
         EJECT
RSETXMS  MODENTRY
         SELECT EVERY
         WHEN  CR,R2,NE,R3
            SSAR  R3
         WHEN  LTR,R4,R4,NZ
            SAC   Ø(R4)
         ENDSEL
         MODEXIT
         EJECT
EXEC_EX  MODENTRY
         MVC   PSFLAGS(8),FLAGS        .BACKUP FLAGS AND XCELL
         LA    R8,XCELL+2
         PERF  EVALBD                  .GET A(DEST INSTRUCTION)
         LAM   R1,R1,=F'Ø'
         CLI   Ø(R1),X'44'             .EX OF AN EX WILL GIVE SØC3 -
         BE    ILGLOP                  .HE'S DOING SOMETHING STUPID
         XR    R15,R15
         IC    R15,Ø(,R1)              .GET EXECUTED OPCODE
         SLL   R15,1                   .* 2
         L     R2,=A(OPFLAGS)
         LH    RØ,Ø(R15,R2)            .EXECUTED CODE'S FLAGS
         STH   RØ,FLAGS
         SELECT
         WHEN  TM,FLAGS+1,RRBIT,O
            MVC   XCELL(2),Ø(R1)
         WHEN  TM,FLAGS+1,SSBIT,O
            MVC   XCELL(6),Ø(R1)
         WHEN  NONE
            MVC   XCELL(4),Ø(R1)
         ENDSEL
         IC    R15,PSXCELL+1           .GET EX-REG
         IF    N,R15,=XL4'FØ',NZ       .NOT 'EX RØ,XXXX'
            SRL   R15,2                .SHIFT TO LO-NIBBLE, * 2
            L     R15,REGTBL(R15)      .GET REG VALUE
            EX    R15,ORI              .OR LOW-BYTE INTO INST
         ENDIF
         MVC   CODEFLD,XCELL           .COPY TO WRK AREA
         PERF  SWITCHER                .DISPATCHER
         PERF  PRT_EX                  .PRINT RESULTS
         IF    TM,PSFLAGS,BRBIT,Z      .WAS A BRANCH INST EX'D?
            L     R9,NEW_IPTR          .NO, SO POINT TO NXT SEQ
            LA    R9,4(,R9)            .INSTRUCTION
         ENDIF
         MODEXIT
         EJECT
PRT_EX   MODENTRY
         IF    CLC,AR_LINE,NE,=CL133' '
```


© 1999. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (940) 455 7050, fax (940) 455 2492.

```
            XC    AR_LINE,PRTLINE
            XC    PRTLINE,AR_LINE
            XC    AR_LINE,PRTLINE
         ENDIF
         MVC   EXD_LINE,PRTLINE        .PRTLINE WAS BUILT BY SWITCHER
         MVC   EXD_LINE(14),=CL14'    EX''D INST:'
         MVI   PRTLINE,X'40'           .CLEAR PRTLINE
         MVC   PRTLINE+1(L'PRTLINE-1),PRTLINE
         XC    FLAGS(8),PSFLAGS        .SWAP FLAGS AND PSFLAGS,
         XC    PSFLAGS(8),FLAGS        .XCELL AND PSXCELL
         XC    FLAGS(8),PSFLAGS
         PERF  SHOWINST                .SHOW INST PTR, INSTR & OP-CODE
         IC    R3,XCELL+1              .SHOW INVOLVED
         PERF  REG_OPS                 .REGISTER
         LA    R3,XCELL+2              .SHOW DESTINATION
         PERF  SHOW_BD                 .INSTR ADDR
         IF    TM,XCELL+1,B'11110000',NZ  .R0 USED FOR EX?
            IC    R3,XCELL+1           .NO - SO DISPLAY THE REG
            PERF  SHOW_GRS
         ENDIF
         SHOW_EFA TO=(EFA1-1),FOR=4,FROM=(XCELL+2)  .DISPLAY ADDR+CONTS
         IF    IAC,R14,NZ
            SHOW_AR FROM=(XCELL+2),TO=(EFA2)
         ENDIF
         PERF  WRITE
         MVC   PRTLINE,EXD_LINE
         PERF  WRITE
         IF    CLC,AR_LINE,NE,=CL133' '
            MVC   PRTLINE,AR_LINE
            PERF  WRITE
         ENDIF
         IF    TM,PSFLAGS+1,LMSTMBIT,O
            PERF  DUMPREGS
         ENDIF
         IF    TM,FLAGS+1,ARBIT,O,OR,       .ACCESS REGISTERS?         +
               CLC,=X'0101',EQ,PSXCELL      .PR?
            PERF  DUMP_ARS
         ENDIF
         MODEXIT
         EJECT
SWITCHER MODENTRY
         SELECT
         WHEN  CLC,=X'0101',EQ,XCELL
            PERF  EXEC_PR
         WHEN  CLC,=X'0102',EQ,XCELL
            PERF  EXEC_UPT
         WHEN  CLI,XCELL,EQ,X'B2'      .B2 EXTENDED OPCODE?
            PERF  EXEC_B2              .YES, BUT NOT DXR (B22D)
         WHEN  CLI,XCELL,EQ,X'A7'      .E7 EXTENDED OPCODE?
            PERF  EXEC_A7              .YES
         WHEN  CLI,XCELL,EQ,X'E5'      .E5 EXTENDED OPCODE?
            PERF  EXEC_E5              .YES
```

© 1999. Reproduction prohibited. Please inform Xephon of any infringement.

```
         WHEN  CLI,XCELL,EQ,X'8Ø'      .SSM (SET SYSTEM MASK)
            PERF  EXEC_SSM             .YES
         WHEN  CLI,XCELL,EQ,X'EE'      .PLO (PERFORM LOCKED OP)
            PERF  EXEC_PLO             .YES
         WHEN  TM,FLAGS,ILGLBIT,O      .ILLEGAL INSTRUCTION(MAY BE THAT
            B     ILGLOP               .TRACE CANNOT HANDLE INSTR.)
         WHEN  TM,FLAGS,FLOATBIT,O     .FLOATING-POINT INSTRUCTION?
            PERF  EXEC_FLT             .YES - INCLUDING DXR (B22D)
         WHEN  CLI,XCELL,EQ,X'47',OR,CLI,XCELL,EQ,7      .BC,BCR
            PERF  EXEC_BC              .BC & BCR
         WHEN  CLI,XCELL,GE,X'84',AND,CLI,XCELL,LE,X'87'
            PERF  EXEC_BX              .BRXH, BRXLE, BXH & BXLE
         WHEN  TM,FLAGS,BRBIT,O
            PERF  EXEC_BR              .ANY INST THAT MAY GEN BRANCH
         WHEN  CLI,XCELL,EQ,X'ØA'
            PERF  EXEC_SVC             .SVC INSTRUCTIONS
         WHEN  CLI,XCELL,EQ,X'9Ø',OR,CLI,XCELL,EQ,X'98'
            PERF  EXEC_LM              .STM OR LM
         WHEN  CLI,XCELL,EQ,X'9A',OR,CLI,XCELL,EQ,X'9B'
            PERF  EXEC_LM              .LAM OR STAM
         WHEN  CLI,XCELL,EQ,X'A8',OR,CLI,XCELL,EQ,X'A9'
            PERF  EXEC_EXTLONG         .MVCLE OR CLCLE
         WHEN  TM,FLAGS+1,SIBIT,O
            PERF  EXEC_SI              .TYPE SI INST.
         WHEN  TM,FLAGS+1,RSBIT,O
            PERF  EXEC_RS              .TYPE RS
         WHEN  TM,FLAGS+1,RRBIT,O
            PERF  EXEC_RR              .TYPE RR
         WHEN  TM,FLAGS+1,SSBIT,O
            PERF  EXEC_SS              .TYPE S-S
         WHEN  NONE
            PERF  EXEC_RX              .NOTHING ELSE, MUST BE R-X
         ENDSEL
         MODEXIT
         EJECT
SHOWINST MODENTRY
         XR    RØ,RØ
         SAR   R1,RØ
         L     R1,NEW_IPTR
         TST31 R1                     .SET HI-ORDER BIT IF AMODE 31
         ST    R1,DUB
         UNPK  I_PTR(9),DUB(5)
         MVI   I_PTR+8,X'4Ø'
         TR    I_PTR,HEXCHAR-C'Ø'
         PERF  PRT_OFST               .SHOW OFFSET FROM CSECT START
         PERF  PRT_HXOP               .SHOW HEX OPCODE+OPERANDS
         XR    R1,R1
         IC    R1,REALCC
         SRL   R1,4
         IC    R1,BCDCC(R1)
         STC   R1,CC                  .SHOW CURRENT CC
         SELECT
```


© 1999. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (940) 455 7050, fax (940) 455 2492.

```
         WHEN  CLC,=X'Ø1Ø1',EQ,XCELL
            MVC   OPCODE,=CL5'PR'
         WHEN  CLI,XCELL,EQ,X'B2'      .EXTENDED OPCODE?
            IC    R1,XCELL+1           .YES, SO GET NAME
            MH    R1,=H'5'             .FROM DIFFERENT TABLE
            A     R1,=A(B2NAMES)
            MVC   OPCODE,Ø(R1)
         WHEN  CLI,XCELL,EQ,X'A7'      .EXTENDED OPCODE?
            IC    R1,XCELL+1           .YES, SO GET NAME
            N     R1,=F'15'            .ONLY LOW-NIBBLE
            MH    R1,=H'5'             .FROM DIFFERENT TABLE
            A     R1,=A(A7NAMES)
            MVC   OPCODE,Ø(R1)
         WHEN  CLI,XCELL,EQ,X'E5'      .EXTENDED OPCODE?
            IC    R1,XCELL+1           .YES, SO GET NAME
            MH    R1,=H'5'             .FROM DIFFERENT TABLE
            A     R1,=A(E5NAMES)
            MVC   OPCODE,Ø(R1)
         WHEN  NONE
            IC    R1,XCELL
            MH    R1,=H'5'
            L     R15,=A(BCDOP)
            LA    R1,Ø(R1,R15)
            MVC   OPCODE,Ø(R1)
         ENDSEL
         IF    CLI,XCELL,EQ,X'47',OR,CLI,XCELL,EQ,7   .BC OR BCR?
            IC    R1,OPCODE+2          .PICK UP 'R' OR BLANK
            SELECT
            WHEN  TM,XCELL+1,B'1111ØØØØ',Z
               MVC   OPCODE,=CL5'NOP'
               STC   R1,OPCODE+3       .STORE 'R' OR BLANK (NOP/NOPR)
            WHEN  CC=1
               MVC   OPCODE,=CL5'B'
               STC   R1,OPCODE+1       .STORE 'R' OR BLANK (B/BR)
            ENDSEL
         ENDIF
         MODEXIT
         EJECT
PRT_OFST MODENTRY            .SHOW OFFSET FROM CSECT START
         IF    ICM,R3,15,CUREP,Z       .UNKNOWN EPA
            MVC   OFFSET(5),=CL5' '
         ELSE
            L     R5,NEW_IPTR
            LR    R2,R5
            IF    S,R5,4(R3),M,OR,C,R5,GT,=F'65535'
               MVC   OFFSET(5),=CL5' '
            ELSE
               ST    R5,DUB
               UNPK  OFFSET(5),DUB+2(3)
               MVI   OFFSET+4,C':'
               TR    OFFSET,HEXCHAR-C'Ø'
            ENDIF
         ENDIF
         MODEXIT
```

© 1999. Reproduction prohibited. Please inform Xephon of any infringement.

```
         SPACE 3
PRT_HXOP MODENTRY
         SELECT
         WHEN  CLI,XCELL,EQ,X'B2'
            UNPK  HEXOP(9),XCELL(5)
            MVI   HEXOP+8,X'40'
            TR    HEXOP(8),HEXCHAR-C'0'
         WHEN  TM,FLAGS+1,RRBIT,O
            UNPK  HEXOP(5),XCELL(3)
            MVI   HEXOP+4,X'40'
            TR    HEXOP(4),HEXCHAR-C'0'
         WHEN  TM,FLAGS+1,SSBIT,O
            UNPK  HEXOP(13),XCELL(7)
            MVI   HEXOP+12,X'40'
            TR    HEXOP(12),HEXCHAR-C'0'
         WHEN  NONE
            UNPK  HEXOP(9),XCELL(5)
            MVI   HEXOP+8,X'40'
            TR    HEXOP(8),HEXCHAR-C'0'
         ENDSEL
         MODEXIT
         EJECT
EVALBD   MODENTRY
         LH    R1,0(,R8)              .R8 POINTS TO BDDD
         LR    R15,R1                 .COPY R1
         N     R1,=F'4095'            .KEEP DDD IN R1
         SELECT EVERY
         WHEN  N,R15,=A(X'F000'),NZ   .BASE REG NE 0?
            SRL   R15,12-2            .SO ADD VALUE OF THAT REG
            A     R1,OLDREGS(R15)
         WHEN  TM,FLAGS+1,RXBIT,O     .RX INSTRUCTION?
            IC    R14,XCELL+1         .YEP, SO TEST IF
            IF    N,R14,=F'15',NZ     .INDEX REGISTER NE R0
               SLL   R14,2            .IT ISN'T, SO ADD THE INDEX REG
               A     R1,OLDREGS(R14)
            ENDIF
         ENDSEL
         LA    R1,0(,R1)              .ZERO HI-ORDER BIT/BYTE,
         MODEXIT
         EJECT
REG_OPS  MODENTRY
         LR    R1,R3
         N     R1,=A(X'F0')
         SRL   R1,4
         CVD   R1,DUB
         OI    DUB+7,X'0F'
         UNPK  FIELDS(3),DUB+6(2)
         MVI   FIELDS,C'R'
         SELECT
         WHEN  CLI,XCELL,EQ,X'B2'
            SELECT
            WHEN  TM,FLAGS+1,B2R2BIT,O
```


© 1999. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (940) 455 7050, fax (940) 455 2492.

```
               MVI   FIELDS+3,C','
               LA    R6,FIELDS+4
               PERF  REG_OP2
            WHEN  TM,FLAGS+1,B2ADRBIT+B2STGBIT,NZ
               MVI   FIELDS+3,C','
               LA    R6,FIELDS+4
            ENDSEL
         WHEN  CLI,XCELL,EQ,X'A7'
            MVI   FIELDS+3,C','
            LA    R6,FIELDS+4
         WHEN  TM,FLAGS+1,RRBIT,O,AND,CLI,XCELL,NE,4,ORIF,              +
               CLI,XCELL,EQ,X'87',OR,CLI,XCELL,EQ,X'86',OR,             +
               CLI,XCELL,EQ,X'BA',OR,CLI,XCELL,EQ,X'BB',OR,             +
               CLI,XCELL,EQ,X'9A',OR,CLI,XCELL,EQ,X'9B',OR,             +
               CLI,XCELL,EQ,X'9Ø',OR,CLI,XCELL,EQ,X'98'
            MVI   FIELDS+3,C','
            LA    R6,FIELDS+4
            PERF  REG_OP2
         WHEN  CLI,XCELL,EQ,X'A8',OR,CLI,XCELL,EQ,X'A9'
            MVI   FIELDS+3,C','
            LA    R6,FIELDS+4
            PERF  REG_OP2
         WHEN  TM,FLAGS+1,RSBIT+RXBIT,NZ
            MVI   FIELDS+3,C','
            LA    R6,FIELDS+4
         ENDSEL
         MODEXIT
         SPACE 3
REG_OP2  MODENTRY
         LR    R1,R3
         N     R1,=F'15'
         CVD   R1,DUB
         OI    DUB+7,X'ØF'
         UNPK  Ø(3,R6),DUB+6(2)
         MVI   Ø(R6),C'R'
         LA    R6,3(,R6)
         MODEXIT
         EJECT
SHOW_BD  MODENTRY
         LH    R1,Ø(,R3)
         N     R1,=F'4Ø95'
         CVD   R1,DUB
         OI    DUB+7,X'ØF'
         UNPK  Ø(4,R6),DUB+5(3)
         LA    R6,4(,R6)
         IF    TM,FLAGS+1,SSBIT,Z,AND,                                  +
               TM,XCELL+1,X'ØF',Z,AND,TM,XCELL+2,X'FØ',Z
               B     BD_RET
         ENDIF
         MVI   Ø(R6),C'('
         IF    TM,FLAGS+1,RXBIT,O
            IF    TM,XCELL+1,B'1111',Z   .X-REG = RØ?
               MVI   1(R6),C','          .SHOW Ø-X WITH DDDD(,B)
```

© 1999. Reproduction prohibited. Please inform Xephon of any infringement.

```
               LA    R6,1(,R6)
            ELSE
               IC    R1,XCELL+1
               N     R1,=F'15'
               CVD   R1,DUB
               OI    DUB+7,X'ØF'
               UNPK  1(3,R6),DUB+6(2)
               MVI   1(R6),C'R'
               LA    R6,4(,R6)
               IF    TM,XCELL+2,B'11110000',Z
                  MVI   Ø(R6),C')'
                  B     BD_RET
               ELSE
                  MVI   Ø(R6),C','
               ENDIF
            ENDIF
         ENDIF
         ICM   R1,3,Ø(R3)
         SRL   R1,12
         CVD   R1,DUB
         OI    DUB+7,X'ØF'
         UNPK  1(3,R6),DUB+6(2)
         MVI   1(R6),C'R'
         MVI   4(R6),C')'
         LA    R6,5(,R6)
BD_RET   DS    ØH
         MODEXIT
         EJECT
SHOW_EFA MODENTRY
         PERF  EVALBD
         TST31 R1
         ST    R1,DUB
         UNPK  1(9,R6),DUB(5)
         TR    1(8,R6),HEXCHAR-C'Ø'
         MVI   Ø(R6),C'('
         MVI   9(R6),C')'
         LA    R2,11(,R6)
         LAM   RØ,R15,=16F'Ø'
         XR    R14,R14
         XR    R15,R15
         IC    R15,Ø(,R8)
         SRL   R15,4
         SLL   R15,2
         LA    R15,AR_OLD(R15)
         IF    CLI,XCELL,EQ,X'B2'
            IC    R14,XCELL+1
            A     R14,=A(AR_B2_ØØ)
            IF    TM,Ø(R14),AR_B2,O,AND,IAC,RØ,NZ
               LAM   R1,R1,Ø(R15)
            ENDIF
         ELSE
            IC    R14,XCELL
            A     R14,=A(AR_ØØ)
```


© 1999. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (940) 455 7050, fax (940) 455 2492.

```
      IF    IAC,R0,NZ
         SELECT
         WHEN  TM,FLAGS+1,RXBIT+RSBIT,NZ
            IF    TM,0(R14),AR_B2,O
               LAM   R1,R1,0(R15)
            ENDIF
         WHEN  TM,FLAGS+1,SIBIT,O
            IF    TM,0(R14),AR_B1,O
               LAM   R1,R1,0(R15)
            ENDIF
         WHEN  TM,FLAGS+1,SSBIT,O
            LA    R0,XCELL+2
            SELECT
            WHEN  CR,R0,EQ,R8
               IF    TM,0(R14),AR_B1,O
                  LAM   R1,R1,0(R15)
               ENDIF
            WHEN  TM,0(R14),AR_B2,O
               LAM   R1,R1,0(R15)
            ENDSEL
         ENDSEL
      ENDIF
   ENDIF
   IF    LTR,R5,R5,NZ
      IF    CLI,XCELL,EQ,X'DB'
         LA    R1,XMS_WRK
      ENDIF
      LR    R3,R5
      DO    WHILE=(C,R3,GT,=F'7')
         UNPK  0(15,R2),0(8,R1)
         LA    R2,14(,R2)
         LA    R1,7(,R1)
         S     R3,=F'7'
      ENDDO
      LR    R14,R3
      BCTR  R14,0
      EX    R14,MOVE_OP
      LR    R14,R3
      SLL   R14,1+4           .*2, AND SHIFT TO NEXT NIBBLE
      LA    R15,0(R3,R14)
      EX    R15,UNPK_OP
      SLL   R3,1
      LA    R14,0(R3,R2)
      MVI   0(R14),X'40'
      SLL   R5,1                    .* 2
      BCTR  R5,0                    .MAKE EXEC LEN
      EX    R5,TRANS
```

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

Pieter Wiid
Advisory Systems Engineer
Perestel (South Africa)

© Xephon 1999

© 1999. Reproduction prohibited. Please inform Xephon of any infringement.

Sterling Software has announced a new version of its Sams:Disk to control Unix resources and data from within OS/390. The OS/390 Unix Edition of Sams:Disk data management tool provides back-up to disk or tape, recovery and reporting on OS/390 Unix files. Users can apply back-up and business continuance policies tailored for the MVS environment to the management of OS/390 Unix files and directories.

It also enables MVS storage personnel to use familiar MVS-based language and procedures to centralize data management operations, promising to cut training costs.

For further information contact:
Sterling Software, 1800 Alexander Bell Drive, Reston, VA 22091, USA.
Tel: (703) 264 8000
Fax: (703) 264 1312
Sterling Software, 1 Longwalk Road, Stockley Park, Uxbridge, Middlesex, UB11 1DB, UK.
Tel: (0181) 867 8000

* * *

Change management outfit, Serena Software, has announced Release 8.2.2 of its Comparex software for OS/390 with a new Euroexit option for conversions from the euro to the local currency unit, or from the local currency unit to the euro.

Comparex performs single-step comparisons of the contents of any two libraries, directories, files, or databases. It is designed to detect differences between files of like and dissimilar content, structure, or record length, and can isolate changes and generate a difference report. Besides the euro option, Release 8.2.2 is designed to improve the ease of use and efficiency of the existing copybook parsing utility. This lets users define the data for comparison by generating keywords and options directly from copybook field definitions. Besides MVS PDSs, users can now directly access CA-Panvalet or CA-Librarian copybooks when using the parsing utility.

For further information contact:
Serena Software International, 500 Airport Blvd, Second Floor, Burlingame, CA 94010-1904, USA.
Tel: (650) 696 1800
Fax: (650) 696 1776

* * *

IBM has announced the first of its enterprise storage resource management (ESRM) products. StorWatch Reporter is for storage asset and capacity management and looks out over an IP network to discover servers attached to the network and determine how much disk filesystem capacity each server has. It builds an inventory of operating system type, version, model level, total disk space capacity in filespaces, and current utilization of that disk space. It gathers this information at intervals specified by the storage administrator, who can look at one consolidated report to see storage usage on OS/390 Unix System Services, AIX, Solaris, HP-UX, IRIX, Windows NT, IntranetWare, and OS/2 servers.

Contact your local IBM representative for further information.