203

MVS

August 2003

update

In this issue

3 IPLing just got easier

8 Possible rounding errors in COBOL on the mainframe

15 Automatically switching prefixes

17 WLM postprocessing made easy

26 Comparing two files

36 VSAM information giving advice for reorganization

46 Universal procedure for compiling and binding Enterprise PL/I programs

59 Extending the standard module design

70 Using CSI to identify VSAM datasets defined with IMBED, REPLICATE, and KEYRANGE

74 MVS news

Published by

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

North American office

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

Subscriptions and back-issues

A year's subscription to *MVS Update*, comprising twelve monthly issues, costs £340.00 in the UK; $505.00 in the USA and Canada; £346.00 in Europe; £352.00 in Australasia and Japan; and £350.00 elsewhere. In all cases the price includes postage. Individual issues, starting with the January 1999 issue, are available separately to subscribers for £29.00 ($43.50) each including postage.

MVS Update on-line

Code from *MVS Update*, and complete issues in Acrobat PDF format, can be downloaded from our Web site at http://www.xephon.com/mvs; you will need to supply a word from the printed issue.

Editor

Trevor Eddolls

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.

Contributions

When Xephon is given copyright, articles published in *MVS Update* are paid for at the rate of £170 ($260) per 1000 words and £100 ($160) per 100 lines of code for the first 200 lines of original material. The remaining code is paid for at the rate of £50 ($80) per 100 lines. In addition, there is a flat fee of £30 ($50) per article. To find out more about contributing an article, without any obligation, please download a copy of our *Notes for Contributors* from www.xephon.com/nfc.

Printed in England.

IPLing just got easier

The questions began when it was discovered that SYS1.IPLPARM had over 100 members. Eighteen were currently in use in regular rotation, as IODF and IOCDS were updated. And the operators had a regularly-updated wall chart with the current and previous IPL parameter value for each of the six LPARs on a single zSeries 900.

With a little help from another new person on-site, SYS1.IPLPARM is now down to one member. Meanwhile, the operators can forget about the IPL parameter value because it can be set once on the HMC and left: it remains constant across all LPARs, even when the IODF changes.

WHERE WE STARTED

The main production system, MVSA, had been using SYS1.IPLPARM member LOAD2A:

```
IODF     Ø2 SYS1     EPRDMVSA ØØ                              NUCLEUS  1
SYSCAT   ASYSL1113CSYS1.PRODPLEX.CATALOG
SYSPARM  (ØØ,SA)
IEASYM   (ØØ,RS)
INITSQA  ØØØØM ØØØ1M
NUCLST   ØØ N
PARMLIB  SYS1.SYSRES.EXTNSION.PARMLIB                  ******
PARMLIB  SYS1.ZOS12.PARMLIB
PARMLIB  SYS1.PARMLIB
```

MVSB, the system used to install and test new software, had been using member LOAD2Z:

```
IODF     Ø2 SYS1     ATSTMVSB ØØ                              NUCLEUS  1
SYSCAT   BSYSL1113CSYS1.MVSB.CATALOG
SYSPARM  (SZ,SB)
NUCLST   ØØ N
IEASYM   (ØØ,RS)
INITSQA  ØØØØM ØØØ1M
PARMLIB  SYS1.SYSRES.EXTNSION.PARMLIB                  ******
PARMLIB  SYS1.ZOS12.PARMLIB
PARMLIB  SYS1.PARMLIB
```

As you can see, the LOAD*xx* members are similar, but not identical. The most obvious difference is on the first line, where the operating system configuration identifier differs for each LPAR – ATSTMVSB versus EPRDMVSA. LOAD1A (not shown) differs from LOAD2A (see above) only by the IODF number specified just after IODF on the first line. If you rotate between three IODFs, you must have three LOAD*xx* members. We have six LPARs, so that is how we got to 18 active LOAD*xx* members.

NEW Z/OS FEATURES

Two features have been added to z/OS in recent years that make it possible to use one LOAD*xx* member instead of 18. Eliminating the specification of the IODF number on the IODF statement is possible with HCD, because it can insert an IODF pointer into the IOCDS. Unfortunately, it does not happen by default because HCD releases the IOCDS build job while it still has the IODF file open. To overcome this problem, be sure to specify TYPRUN=HOLD on the JOB card that you create for HCD just before submitting the IOCDS build batch job. Exit HCD and then release the job.

Instead of specifying the IODF number, there are a number of different symbols you can specify. I recommend equals signs (==) because a Wait state is forced if the IODF pointed to by the IOCDS does not exist.

The second new feature is the ability to specify statements specific to each LPAR in a single LOAD*xx* member. The LPARNAME statement provides this capability.

JUST ONE IPLPARM MEMBER

Put it all together and here is what you get as LOAD00, which works for all IODFs and all LPARs:

```
NUCLEUS  1                                                SYSCAT
ASYSL1113CSYS1.PRODPLEX.CATALOG
IEASYM   (00,RS)
INITSQA  0000M 0001M
```

```
NUCLST   ØØ N
PARMLIB  SYS1.SYSRES.EXTNSION.PARMLIB                  ******
PARMLIB  SYS1.ZOS12.PARMLIB
PARMLIB  SYS1.PARMLIB
LPARNAME EPRDMVSA
IODF     == SYS1     EPRDMVSA ØØ
SYSPARM  (ØØ,SA)
LPARNAME APRDMVSE
IODF     == SYS1     APRDMVSE ØØ
SYSPARM  (ØØ,SE)
LPARNAME CPRDMVSH
IODF     == SYS1     CPRDMVSH ØØ
SYSPARM  (ØØ,SH)
LPARNAME APRDMVSI
IODF     == SYS1     APRDMVSI ØØ
SYSPARM  (ØØ,L)
SYSCAT   ISYSL1113CSYS1.MVSI.CATALOG
LPARNAME ATSTMVSB
IODF     == SYS1     ATSTMVSB ØØ
SYSPARM  (SZ,SB)
SYSCAT   BSYSL1113CSYS1.MVSB.CATALOG
```

The statements common to all LPARs are included first, then those for each LPAR. Note how statements that can be specified once, such as SYSCAT, can be overridden for a specific LPAR. Finally, if you are wondering about the six asterisks to the far right of the first PARMLIB in all the LOAD*xx* members shown, that indicates that the parmlib is located on the system residence volume; change system residence volumes and you are using a different but identically-named parmlib.

DISPLAY IPLINFO

Not sure what LOAD*xx* member or IODF is in use? The console command Display IPLINFO provides a lot of useful information. On MVSB, with the new LOAD00 in place, here is what you would see:

```
IEA63ØI  OPERATOR E667800  NOW ACTIVE,   SYSTEM=MVSB    , LU=N11521A D
IPLINFO
IEE254I  15.19.27 IPLINFO DISPLAY Ø31
 SYSTEM IPLED AT Ø6.35.58 ON Ø4/11/2ØØ3
 RELEASE z/OS   Ø1.Ø2.ØØ
 USED LOADØØ IN SYS1.IPLPARM ON 45DF
 ARCHLVL = 2   MTLSHARE = N
```

```
 IEASYM LIST = (ØØ,RS)
 IEASYS LIST = (SZ,SB) (OP)
 IODF DEVICE 45DF
 IPL DEVICE 491B VOLUME OS39ØM
```

Before the change, here is what it looked like on MVSA:

```
D IPLINFO                                  IEE254I  15.22.Ø3 IPLINFO
DISPLAY 921
 SYSTEM IPLED AT 16.26.18 ON Ø4/13/2ØØ3
 RELEASE z/OS   Ø1.Ø2.ØØ
 USED LOAD2A IN SYS1.IPLPARM ON 45DF
 ARCHLVL = 2   MTLSHARE = N
 IEASYM LIST = (ØØ,RS)
 IEASYS LIST = (ØØ,SA) (OP)
 IODF DEVICE 45DF
 IPL DEVICE 4Ø2A VOLUME ZOSØØ2
```

THE CHANGE REQUEST

Implementation of this change – having just one member (LOAD00) in SYS1.IPLPARM – required a change request. Here are the implementation and backout instructions given to data centre staff to make this change during a scheduled IML.

Implement

Set IOCDS – Change IOCDS to A0.

On the HMC, double-click *Groups* then double-click *Defined CPCs* then double-click the *A20641C4* icon. Unlock (set Lockout Disruptive Tasks to No) and then hit the *Change Options* button. Select PORA0 as the Profile Name (from the list). Push the two *Save* buttons that appear.

Activate

On the HMC, single-click on the A20641C4 icon, and double-click on the *Activate* icon. (To see the Activate icon, you may have to repeatedly click on the *Rotate* icon (circular arrow) in the bottom-right corner of the screen.)

IPL each system with the new standard IPL parameters – 45DF00 for all systems.

Backout

Set IOCDS – Change IOCDS to A2.

On the HMC, for A20641C4, change the profile to DEFAULT (see Implementation Plan for details) and Activate.

Re-IML.

Re-IPL each system with the previous IPL parameters:

MVSA – 45DF2A

MVSE – 45DF2E

MVSH – 45DF2H

MVSI – 45DF2i.

OTHER CHANGES

Another improvement was implemented at the same time. As shown in the first step of *Implement*, the PORA0 IML profile had been previously created with A0 specified as the IOCDS. Previously, the operators had to change the IOCDS number in the DEFAULT IMS profile. PORA1 and PORA2 were also created for future IMLs when the IOCDS is changed.

The final change made was to keep the IOCDS and IODF numbers in sync. Now, IOCDS A0 is always used with IODF00, A1 with IODF01, and A2 with IODF02. A three-way rotation seemed more than adequate.

NEXT

What is next? We hope to substantially reduce the number of LPARs. Historically, they had been created to resolve performance problems. But those are now being addressed with a complete, from-scratch, redesign of WLM settings. Preliminary results are very encouraging.

Both Cheryl Watson and the Washington System Centre (WSC) provide starting points for WLM settings. WSC was chosen, with

test results reviewed by SHARE speaker Tom Russell of IBM, and his suggested improvements implemented.

Who knows? We may even be able to repeat last month's processor downgrade, and move down yet another level of cost.

Jon E Pearkins
Adiant Corporation (Canada)

Possible rounding errors in COBOL on the mainframe

INTRODUCTION

On our site the majority, if not all, of our programs are developed in a Windows client/server environment. Most of these programs are used in the client/server world for online transactions. The batch environment is made up of transferred source recompiled on the mainframe.

Recently, by coincidence, it was discovered that, for the exact same time period, there was a discrepancy between the computed value in the client/server world and that of the mainframe. This was at first difficult to understand because the source was the same, it could only be down to differences in interpretation in the different environments.

After much searching, the location of the error was discovered to be rounding errors in the compiled COBOL on the mainframe.

PROBLEM

Various variables in the system are stored in one form and then used in this form or a derivative of this form. For example, a percentage would be stored as 12.5 but then used as 0.125. This method of storage and usage is also used by us to work to significant decimal places. To enable this, the COMPUTE

ROUNDED statement is used. It is used in such a way as to shift variable contents significant positions right or left. This is done by dividing or multiplying by an exponential expression of 10 (the factor being a whole number), eg:

```
COMPUTE A ROUNDED = B / (10 ** factor)
```

where *factor* is a whole number.

When B had a value of 9.9875 this was rounded to 9.988 in the client/server environment and truncated to 9.987 on the mainframe. This rounding error, through future multiplication, produced a difference and a loss for us of about 10 pence on a quarter yearly insurance policy.

Further investigation has determined when the exponential expression *factor* is defined with a decimal point, eg FACTOR PIC 9V9, then this rounding error occurs when the digit after the last significant digit is 5 (0 to 4 truncated as expected, 6 to 9 rounded as expected, but 5 is truncated, instead of rounded).

In this article I have included a test program to allow the readers to determine whether this problem could also occur at their site. The results when displayed should, in an error-free case, all be the same – that is 9.98 (0.00000998) and not as we have experienced with the occasional 9.97 (0.00000997).

CONCLUSION

This problem may be unique to us. I have documented it and have reported it to IBM. The problem occurs when we use either the COBOL for MVS V2R2 compiler or the newer Enterprise COBOL V3R2.

The discrepancy may not appear to be much at first, but it brings with it other problems that need to be addressed:

- If it is left, it will, every now and again, create a discrepancy in the accounting system, which will need to be documented.

  We can get over the problem by making a MOVE to a factor variable defined without a decimal point (we know that it is

always a whole number at our site). The change however will probably have the effect that somewhere else in the system a balancing discrepancy would then occur and have to be accounted for.

- If we simply change the code, the subsequent change in a customer's premiums could result in their being able to cancel their policy because of an unannounced premium increase.
- If it is an IBM problem, then a fix could cause similar problems to our work-around and perhaps can be applied only between years to avoid accounting discrepancies for a specific accounting period.

A test program and results follow (note: the test program requires no input).

TEST PROGRAM SOURCE CODE

```
IDENTIFICATION DIVISION.
*****************************************************************
 PROGRAM-ID. COMPUØØ1.
*================================================================
 ENVIRONMENT DIVISION.
*================================================================
*****************************************************************
 CONFIGURATION SECTION.
*****************************************************************
 SPECIAL-NAMES.
     DECIMAL-POINT IS COMMA.
*
*================================================================
 DATA DIVISION.
*================================================================
*
*****************************************************************
 WORKING-STORAGE SECTION.
 Ø1 TEST-VARIABLES.
    Ø5 IVAR-1            PIC S9(1Ø)V9(8).
    Ø5 IVAR-2            PIC S9(1Ø)V9(8).
    Ø5 OVAR-1            PIC S9(1Ø)V9(8).
    Ø5 OVAR-2            PIC S9(1Ø)V9(8).
    Ø5 OVAR-3            PIC S9(1Ø)V9(8).
    Ø5 OVAR-4            PIC S9(1Ø)V9(8).
    Ø5 OVAR-5            PIC S9(1Ø)V9(8).
```

```
    Ø5 OVAR-6           PIC S9(1Ø)V9(8).
    Ø5 OVAR-7           PIC S9(1Ø)V9(8).
    Ø5 OVAR-8           PIC S9(1Ø)V9(8).
    Ø5 OVAR-9           PIC S9(1Ø)V9(8).
    Ø5 OVAR-1Ø          PIC S9(1Ø)V9(8).
    Ø5 OVAR-11          PIC S9(1Ø)V9(8).
    Ø5 OVAR-12          PIC S9(1Ø)V9(8).
    Ø5 FACTOR-1         PIC S9(1Ø)V9(8).
    Ø5 FACTOR-2         PIC 9.
    Ø5 FACTOR-3         PIC 9V9.
    Ø5 FACTOR-4         PIC S9(4).
    Ø5 D-IVAR-1         PIC -Z(9)9,9(8).
    Ø5 D-IVAR-2         PIC -Z(9)9,9(8).
    Ø5 D-OVAR-1         PIC -Z(9)9,9(8).
    Ø5 D-OVAR-2         PIC -Z(9)9,9(8).
    Ø5 D-OVAR-3         PIC -Z(9)9,9(8).
    Ø5 D-OVAR-4         PIC -Z(9)9,9(8).
    Ø5 D-OVAR-5         PIC -Z(9)9,9(8).
    Ø5 D-OVAR-6         PIC -Z(9)9,9(8).
    Ø5 D-OVAR-7         PIC -Z(9)9,9(8).
    Ø5 D-OVAR-8         PIC -Z(9)9,9(8).
    Ø5 D-OVAR-9         PIC -Z(9)9,9(8).
    Ø5 D-OVAR-1Ø        PIC -Z(9)9,9(8).
    Ø5 D-OVAR-11        PIC -Z(9)9,9(8).
    Ø5 D-OVAR-12        PIC -Z(9)9,9(8).
    Ø5 D-FACTOR-1       PIC -Z(9)9,9(8).
    Ø5 D-FACTOR-2       PIC 9.
    Ø5 D-FACTOR-3       PIC 9,9.
    Ø5 D-FACTOR-4       PIC -Z(3)9.
*****************************************************************
 LINKAGE SECTION.
*****************************************************************
*===============================================================
 PROCEDURE DIVISION.
*===============================================================
*****************************************************************
 MAIN SECTION.
*****************************************************************
          PERFORM ROUNDING-Ø1
          GOBACK.
     EXIT.
*****************************************************************
*
*****************************************************************
 ROUNDING-Ø1 SECTION.
*****************************************************************
*       ************************************************
*       INITIALISE AND DISPLAY
*       ************************************************
        MOVE 9,975            TO IVAR-1
```

```
           MOVE 2                TO IVAR-2
           DISPLAY ' ** INITIAL VALUES ** '
           DISPLAY 'IVAR-1 = ' IVAR-1
           DISPLAY 'IVAR-2 = ' IVAR-2
      *    ************************************************
      *    COMPUTE ROUNDED
      *    ************************************************
      *    ************************************************
      *    TEST CASE 1
      *    ************************************************
           COMPUTE FACTOR-4 = 8 - IVAR-2
           COMPUTE OVAR-1 ROUNDED =
                   IVAR-1 / (1Ø ** FACTOR-4)
      *    ************************************************
      *    TEST CASE 2
      *    ************************************************
           COMPUTE OVAR-3 ROUNDED =
                   IVAR-1 / (1Ø ** (8 - IVAR-2))
      *    ************************************************
      *    TEST CASE 3
      *    ************************************************
           COMPUTE OVAR-5 ROUNDED =
                   IVAR-1 / (1Ø ** 6)
      *    ************************************************
      *    TEST CASE 4
      *    ************************************************
           COMPUTE FACTOR-1 = 8 - IVAR-2
           COMPUTE OVAR-7 ROUNDED =
                   IVAR-1 / (1Ø ** FACTOR-1)
      *    ************************************************
      *    TEST CASE 5
      *    ************************************************
           COMPUTE FACTOR-2 = 8 - IVAR-2
           COMPUTE OVAR-9 ROUNDED =
                   IVAR-1 / (1Ø ** FACTOR-2)
      *    ************************************************
      *    TEST CASE 6
      *    ************************************************
           COMPUTE FACTOR-3 = 8 - IVAR-2
           COMPUTE OVAR-11 ROUNDED =
                   IVAR-1 / (1Ø ** FACTOR-3)
      *    ************************************************
      *    COMPUTE BACK
      *    ************************************************
           COMPUTE OVAR-2 =
                   OVAR-1 * (1Ø ** FACTOR-4)
           COMPUTE OVAR-4 =
                   OVAR-3 * (1Ø ** (8 - IVAR-2))
           COMPUTE OVAR-6 =
                   OVAR-5 * (1Ø ** 6)
```

```
           COMPUTE OVAR-8 =
                   OVAR-7 * (1Ø ** FACTOR-1)
           COMPUTE OVAR-1Ø =
                   OVAR-9 * (1Ø ** FACTOR-2)
           COMPUTE OVAR-12 =
                   OVAR-11 * (1Ø ** FACTOR-3)
      *    ************************************************
      *    DISPLAY AFTER
      *    ************************************************
           MOVE IVAR-1 TO D-IVAR-1
           MOVE IVAR-2 TO D-IVAR-2
           MOVE OVAR-1 TO D-OVAR-1
           MOVE OVAR-2 TO D-OVAR-2
           MOVE OVAR-3 TO D-OVAR-3
           MOVE OVAR-4 TO D-OVAR-4
           MOVE OVAR-5 TO D-OVAR-5
           MOVE OVAR-6 TO D-OVAR-6
           MOVE OVAR-7 TO D-OVAR-7
           MOVE OVAR-8 TO D-OVAR-8
           MOVE OVAR-9 TO D-OVAR-9
           MOVE OVAR-1Ø TO D-OVAR-1Ø
           MOVE OVAR-11 TO D-OVAR-11
           MOVE OVAR-12 TO D-OVAR-12
           MOVE FACTOR-1 TO D-FACTOR-1
           MOVE FACTOR-2 TO D-FACTOR-2
           MOVE FACTOR-3 TO D-FACTOR-3
           MOVE FACTOR-4 TO D-FACTOR-4
           DISPLAY ' **** RETAINED INPUT VALUES ****** '
           DISPLAY 'IVAR-1 = ' D-IVAR-1
           DISPLAY 'IVAR-2 = ' D-IVAR-2
           DISPLAY ' ******* RESULTS ******** '
           DISPLAY 'TEST CASE 1'
           DISPLAY '----------'
           DISPLAY 'OVAR-1= ' D-OVAR-1
           DISPLAY 'OVAR-2= ' D-OVAR-2
           DISPLAY 'TEST CASE 2'
           DISPLAY '----------'
           DISPLAY 'OVAR-3 = ' D-OVAR-3
           DISPLAY 'OVAR-4 = ' D-OVAR-4
           DISPLAY 'TEST CASE 3'
           DISPLAY '----------'
           DISPLAY 'OVAR-5 = ' D-OVAR-5
           DISPLAY 'OVAR-6 = ' D-OVAR-6
           DISPLAY 'TEST CASE 4'
           DISPLAY '----------'
           DISPLAY 'OVAR-7 = ' D-OVAR-7
           DISPLAY 'OVAR-8 = ' D-OVAR-8
           DISPLAY 'TEST CASE 5'
           DISPLAY '----------'
           DISPLAY 'OVAR-9 = ' D-OVAR-9
```

```
           DISPLAY 'OVAR-1Ø= ' D-OVAR-1Ø
           DISPLAY 'TEST CASE 6'
           DISPLAY '----------'
           DISPLAY 'OVAR-11= ' D-OVAR-11
           DISPLAY 'OVAR-12= ' D-OVAR-12
           DISPLAY '----------'
           DISPLAY 'FACTOR-1 = ' D-FACTOR-1
           DISPLAY 'FACTOR-2 = ' D-FACTOR-2
           DISPLAY 'FACTOR-3 = ' D-FACTOR-3
           DISPLAY 'FACTOR-4 = ' D-FACTOR-4
       EXIT.
*
*
```

OUTPUT FROM TEST PROGRM

```
** INITIAL VALUES **
IVAR-1 = ØØØØØØØØ99750000ä
IVAR-2 = ØØØØØØØØ20000000ä
 **** RETAINED INPUT VALUES ******
IVAR-1 =           9,97500000
IVAR-2 =           2,00000000
 ******* RESULTS ********
TEST CASE 1
----------
OVAR-1=           Ø,00000998
OVAR-2=           9,98000000
TEST CASE 2
----------
OVAR-3 =           Ø,00000997
OVAR-4 =           9,97000000
TEST CASE 3
----------
OVAR-5 =           Ø,00000998
OVAR-6 =           9,98000000
TEST CASE 4
----------
OVAR-7 =           Ø,00000997
OVAR-8 =           9,97000000
TEST CASE 5
----------
OVAR-9 =           Ø,00000998
OVAR-1Ø=           9,98000000
TEST CASE 6
----------
OVAR-11=           Ø,00000997
OVAR-12=           9,97000000
----------
FACTOR-1 =           6,00000000
```

```
FACTOR-2 = 6
FACTOR-3 = 6,Ø
FACTOR-4 =     6
```

Rolf Parker
Systems Programmer (Germany)

Automatically switching prefixes

To know your actual PROFILE prefix, you only have to do a TSO PROFILE LIST request. Sometimes, you may need to switch PROFILE status from the actual value (eg NOPREFIX) to PROFILE PREFIX(userid), or *vice versa*.

Remember: when PROFILE is set to PREFIX(userid), you must specify 'datasetnames' in ISPF browse or edit and in TSO REXX/CLIST instructions (eg ALLOC) using quotes because the prefix will be automatically added in front of all unquoted requests. So if you want to modify another existing CLIST, you'll find it useful to make 'prefixed' or 'noprefixed' the dataset requests just by inserting a line with **%PROL**.

If you require a display before proceeding, type **TSO %PROL AUTOSW(N),** and you will be prompted to change (or not) the actual profile value.

PROL CLIST

```
PROC Ø AUTOSW(Y) DEBUG
/*- SETUP FOR DEBUG IF REQUESTED ----------------------------------*/
   CONTROL MSG LIST NOFLUSH END(ENDO) NOCONLIST NOPROMPT
  IF &DEBUG  =  DEBUG THEN +
   CONTROL MSG LIST NOFLUSH END(ENDO) PROMPT SYMLIST CONLIST
/*- END OF SETUP ---------------------------------------------------*/
/* . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . */
/*                                                                 */
/* PROL: TO HELP YOU KNOW OR CHANGE YOUR TSO PROFILE PREFIX.       */
/*       PROFILE PREFIX IS USUALLY SET TO YOUR USERID,             */
/*       BUT SOMETIMES IT IS USEFUL TO SET IT TO 'NOPREFIX',       */
/*       SO YOU MAY SWITCH BETWEEN PREFIX AND NOPREFIX.            */
/*                                                                 */
```

```
/* USE:  TSO %PROL AUTOSW(N)   IF YOU WISH TO BE PROMPTED           */
/*       DEFAULT: AUTOSW(Y)                                        */
/*                                                                 */
/* . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . */
  CONTROL MSG LIST
/* . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . */
/*                                                                 */
/* BE SURE 'CONTROL MSG LIST' IS BEEN ACTIVATED,                   */
/* OTHERWISE NO TRAPPING OF PROFILE LIST OUTPUT WILL BE DONE       */
/* AND NO MESSAGE WILL BE DISPLAYED AT THE END OF THE CLIST.       */
/* . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . */
  SET &SYSOUTTRAP=1Ø
            PROFILE LIST
  SET &SYSOUTTRAP=Ø
   SET PRFX=&STR(&SYSOUTLINE2)
   SET &L = &SYSINDEX(PREFIX(,&STR(&PRFX))
   IF &L NE Ø THEN DO
      SET &PRFX=&SUBSTR(&L+7:&LENGTH(&PRFX)-1,&STR(&PRFX))
      IF &AUTOSW=Y THEN GOTO Y1
 MES1: +
            WRITE YOUR ACTUAL PROFILE PREFIX IS &PRFX
            WRITE DO YOU LIKE TO SET IT TO NOPREFIX?
            WRITENR (Y/N)
        READ &AUTOSW
 Y1: +
        SELECT &AUTOSW
        WHEN(Y) PROFILE NOPREFIX
        WHEN(N) GOTO ESCI
        OTHERWISE GOTO MES1
        ENDO
                ENDO
   ELSE DO
        SET PRFX=N
      IF &AUTOSW=Y THEN GOTO Y2
 MES2: +
            WRITE YOUR ACTUAL PROFILE IS NOPREFIX
            WRITE DO YOU LIKE TO CHANGE IT TO YOUR USERID (&SYSUID)?
            WRITENR (Y/N)
        READ &AUTOSW
 Y2: +
        SELECT &AUTOSW
        WHEN(Y) PROFILE PREFIX(&SYSUID)
        WHEN(N) GOTO ESCI
        OTHERWISE GOTO MES2
        ENDO
      ENDO
ESCI: +
  EXIT CODE(Ø)
```

Alberto Mungai
Senior Systems Programmer (Italy)

WLM postprocessing made easy

INTRODUCTION

As is commonly known, beginning with z/OS V1R3, compatibility-mode is no longer available and an IPLed system will run in WLM goal-mode only. This means that each installation will be required to have a service definition installed and a WLM policy activated. Once a service definition is in place and the system is running in goal-mode, performance analysts are faced with the task of trying to understand what is going on in the system.

On the other hand, it may happen that, even though an installation is running in goal-mode for quite some time and everything is performing quite well, there are still changes, such as workload, software, or hardware changes, that should cause one to review, re-evaluate, and perhaps to modify WLM goals. This is where using a performance reporter product can be useful.

One of the best ways to review the WLM performance metrics is through the use of both real-time monitors (RMF Monitor III SYSSUM report, for example) and a postprocessor.

There are several areas one will want to look at to quickly gain knowledge of how a given workload is performing in relation to the goals that have been set in the service policy. One should keep in mind the fact that WLM uses three primary metrics to define how it should manage workloads – importance levels, service objectives, and performance index (PI):

- Importance level identifies the service classes according to the order in which WLM is to try to satisfy stated objectives, ie the order they should receive/donate resources. Since WLM dynamically adjusts the resources, the importance level determines how those adjustments are to be made and in what order. It was noticed that there is a strong temptation to place too many units of work into the upper importance levels and thus to overload WLM's decision-making capability.

The consequence of this is that a high importance-level workload that fails to meet its objectives will invariably prevent lower importance level work from being examined.

- The defined service objectives categories (response time, response time percentile, velocity, discretionary) are telling WLM what the standard of measure will be.
- Performance index (PI) is used to evaluate how well the stated objectives are being met. WLM uses performance index in conjunction with importance level to determine what action (if any) should be taken. Most people are aware that a PI greater than 1 indicates that goals are not being met, while a PI of less than 1 indicates they are being exceeded.

Once the work has been classified into service classes with defined goals, the performance index tells us whether the workload on the system is meeting its WLM policy-defined goals, or that nothing is acting in the way we thought it would. Besides the obvious point that a service class is missing its objectives, this may also indicate that a particular objective is simply too aggressive for WLM to ever satisfy. What if you have noticed that a service class isn't performing well because of a WLM-managed resource such as CPU, or that a Sysplex performance index is significantly less than a local performance index? For those performance analysts who have a grasp of Workload Manager concepts, constraints, and analysis techniques, the WLM postprocessor can significantly reduce the time required to perform daily analysis of system performance.

COLLECTING WLM DATA

As mentioned above, Workload Manager periodically assesses the performance of each service class period by comparing the performance achieved by the service class period against the performance goals defined for the service class period. WLM does this by sampling the state of the service class four times per second. This assessment is done at each goal importance level. In this way, WLM can determine whether the service class is

using resources or whether the service class is being delayed in a manner that may be adjustable. These sorts of delay, over which WLM can exert no control, are discarded in this assessment and do not contribute directly to WLM's decision making. Idle time periods are excluded from the samples collection.

In order to document its decisions, WLM creates several SMF records (type 99) for each policy interval, or approximately once every 10 seconds. They can be useful in analysing and understanding the performance characteristics of a site's workload. The records contain performance data for each service class period, a trace of SRM actions, the data SRM used to decide which actions to take, and the internal controls SRM uses to manage work. This can help the performance analyst to determine in detail what SRM is doing to meet workload goals defined with respect to other work, and the types of delays the work is experiencing.

Before proceeding any further it might be helpful to clarify the difference between SMF record type 72 (RMF Workload Activity) subtype 3 and record type 99 (System Resource Manager Decisions) subtype 6 since these two do overlap in some of their content. However, they have two significant differences.

Record type 99(6) contains local and Sysplex-level performance index values as calculated at policy adjustment time (in fact, this subtype contains no new data – everything in it is already in other subtypes of the type 99, but the new record compacts the needed data in one subtype so that one can afford to write that subtype 6 record and can suppress all other subtypes to reduce data volume).

The record type 72(3) does not include a PI value but does contain all the data needed to calculate an average PI for the RMF recording interval. Furthermore, type 99(6) provides the data on WLM's internally-used dynamic service classes, but it is only type 72(3) that contains resource consumption data.

A detailed description of the layout of SMF type 99 record and its subtypes can be obtained from the *MVS System Management*

Facilities (SMF) - SA22-7630-03 manual.

For information about how to use type 99, see *z/OS MVS Programming: Workload Management Services*.

For information about workload management, see *z/OS MVS Planning: Workload Management*.

The mapping macro, IRASMF99, for this record is supplied in SYS1.AMODGEN.

Because SMF type 99 records are written approximately every 10 seconds, one should write them only for certain time periods and define them (in SMFPRM*xx* member) like:

```
SYS(NOTYPE(99))
SYS(TYPE(99(6))
```

CODE

In order to provide a starting point from which one can begin to gather information about the system, an example of the WLM postprocessor JCL statements is included below.

The code is a three-part stream. In the first part (COPY996) selected SMF records (selection being defined by INCLUDE's condition) are copied from the SMF dataset to a VB file, which can be used as a base of archived records. In the second part (WLM99), the captured records are being formatted by invoking REXX EXEC (WLMPP). In the last part (RPT996), the formatted records are being read and a report produced. The field reformatting capability of DFSORT's ICETOOL was used to produce a report from the WLMPP output. For each service class, related information is produced – class name, period, local and Sysplex performance index, goal type defined and goal value measured, period importance, goal percentile, dispatching and I/O priority.

This job stream can be used to create a flexible report of those metrics that can quickly provide us, at a glance, with data about service classes which are and are not meeting specified goals,

local and Sysplex-level PI. From a WLM perspective, a daily or weekly review of the reports should be used to provide a set of measurements to track and provide information for trend analysis. One should choose a busy time frame (1-3 hours) to use as a measurement period for this purpose.

```
//COPY996  EXEC PGM=ICETOOL
//TOOLMSG  DD SYSOUT=*
//DFSMSG   DD SYSOUT=*
//RAWSMF   DD DSN=hlq.SMFDUMPW,DISP=SHR
//SMF99    DD DSN=your.copied.by.sort.to.VB.smf.dataset,
//         SPACE=(CYL,(1)),UNIT=SYSDA,
//         DISP=(NEW,PASS),
//         DCB=(RECFM=VB,LRECL=32756,BLKSIZE=32760)
//TOOLIN   DD *
  COPY FROM(RAWSMF) TO(SMF99) USING(SMFI)
//SMFICNTL DD *
  OPTION SPANINC=RC4,VLSHRT
  INCLUDE COND=(6,1,BI,EQ,99,AND,23,2,BI,EQ,6)
/*
//WLM99  EXEC PGM=IKJEFT01,REGION=0M,DYNAMNBR=50,PARM='%WLMPP'
//SYSEXEC  DD DISP=SHR,DSN=your.rexx.library
//SMF      DD DISP=(SHR,PASS),DSN=your.copied.by.sort.to.VB.smf.dataset
//OUT99    DD DSN=sysuid.output.dataset,
//         SPACE=(CYL,(30,15)),UNIT=SYSDA,
//         DISP=(NEW,PASS),
//         DCB=(RECFM=FB,LRECL=140)
//SYSPRINT DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD DUMMY
/*
//RPT996   EXEC PGM=ICETOOL,REGION=0M
//TOOLMSG  DD SYSOUT=X
//DFSMSG   DD SYSOUT=X
//REPORT   DD SYSOUT=X
//OUT99    DD DISP=(SHR,KEEP),DSN=sysuid.output.dataset
//TEMP     DD DSN=&&TEMPV,SPACE=(CYL,(15,15)),UNIT=SYSDA
//TOOLIN   DD *
  COPY FROM(OUT99) TO(TEMP) USING(SMFI)
  DISPLAY FROM(TEMP) LIST(REPORT)    -
  TITLE('WLM POSTPROCESSOR REPORT') DATE TIME -
   HEADER('TIME')       ON(13,8,CH)    -
   HEADER('SID')        ON(23,4,CH)    -
   HEADER('S.CLASS')    ON(46,8,CH)    -
   HEADER('PERIOD.')    ON(55,1,CH)    -
   HEADER('LOCAL PI')   ON(57,3,CH)    -
   HEADER('SYSPLEX PI') ON(62,3,CH)    -
   HEADER('GOAL TYPE')  ON(67,15,CH)   -
   HEADER('GOAL VALUE') ON(83,3,CH)    -
```

```
   HEADER('IMP')        ON(87,1,CH)    -
   HEADER('PERC.')      ON(90,2,CH)    -
   HEADER('CPU DP')     ON(93,3,CH)    -
   HEADER('IO DP')      ON(97,3,CH)    -
   BREAK(1,11,CH)                      -
   BTITLE('DAILY REPORT')              -
   BLANK
/*
//SMFICNTL DD *
* Example: select only peak period (ie 10 am – 2.pm)
* In a similar fashion one may construct customized INCLUDE statement
* and select other fields (ie LPI > 1, LPI > SPI…)
   OPTION COPY
   INCLUDE COND=(13,5,CH,GT,C'10:00',AND,13,5,CH,LT,C'13:59')
/*
```

WLMPP EXEC

```
/* REXX EXEC to read and format SMF records  */
ADDRESS TSO

 'EXECIO * DISKR SMF ( STEM x. FINIS'
    do i = 1 to x.0

  smftype  = c2d(SUBSTR(x.i,2,1))              /* SMF record type */
  smfstype = c2d(SUBSTR(x.i,19,2))             /* Record subtype  */
  /*-------------------------------------------------------------*/
  /* Check SMF record type & subtype (ie 99.6)                   */
  /*-------------------------------------------------------------*/
  IF smftype = '99' & smfstype = '6' THEN
  DO
    offset   = c2d(SUBSTR(x.i,69,4))  /* Offset to period section  */
    len      = c2d(SUBSTR(x.i,73,2))  /* Length of period section  */
    cpon     = c2d(SUBSTR(x.i,75,2))  /* Number of period sections */
  /*-------------------------------------------------------------*/
  /* Unpack SMF date & decode SMF time                           */
  /*-------------------------------------------------------------*/
    smfdate  = SUBSTR(c2x(SUBSTR(x.i,7,4)),3,5) /* unpack SMF date */
    time     = c2d(SUBSTR(x.i,3,4))             /* decode SMF time */
    time1    = time % 100
    hh       = time1 % 3600
    hh       = RIGHT("0"||hh,2)
    mm       = (time1 % 60) - (hh * 60)
    mm       = RIGHT("0"||mm,2)
    ss       = time1 - (hh * 3600) - (mm * 60)
    ss       = RIGHT("0"||ss,2)
    smftime  = hh||":"||mm||":"||ss             /* Compose SMF time*/
  /*-------------------------------------------------------------*/
  /* Process all class periods                                   */
```

```
/*-------------------------------------------------------------*/
   do j = 0 to cpon
     incr = (offset + (j*len)) - 3      /* Incremental position  */
     sclass = SUBSTR(x.i,incr,8)        /* Class name            */
     period = c2d(SUBSTR(x.i,incr+8,2)) /* Class period number   */

      if period >  '0' then do
       sysid    = SUBSTR(x.i,11,4)      /* System identification */
       syslvl   = SUBSTR(x.i,53,8)      /* System level          */
       sysname  = SUBSTR(x.i,61,8)      /* System name           */
       gt       = c2d(SUBSTR(x.i,incr+10,1)) /* Goal type        */
       pct      = c2d(SUBSTR(x.i,incr+11,1)) /* Goal percentile  */
/*-------------------------------------------------------------*/
/* Reformat goal type values into goal description             */
/*-------------------------------------------------------------*/
        SELECT
          when gt=0 then goal='System/STC/Srv '
          when gt=1 then goal='Shr.Resp (sec.)'
          when gt=2 then goal='Lng.Resp (sec.)'
          when gt=3 then goal='Velocity (%)   '
          when gt=4 then goal='Discretionary  '
        END

       gval  = c2d(SUBSTR(x.i,incr+20,4)) /* Goal value           */
       imp   = c2d(SUBSTR(x.i,incr+24,2)) /* Period importance    */
       dp    = c2d(SUBSTR(x.i,incr+26,1)) /* Dispatching prty.    */
       iodp  = c2d(SUBSTR(x.i,incr+27,1)) /* I/O priority         */
       mpli  = c2d(SUBSTR(x.i,incr+28,2)) /* MPL in-target        */
       mplo  = c2d(SUBSTR(x.i,incr+30,2)) /* MPL out-target       */
       rua   = c2d(SUBSTR(x.i,incr+32,4)) /* Number of ready ASIDs*/
       pspt  = c2d(SUBSTR(x.i,incr+36,4)) /* Time swapped out     */
       psitar= c2d(SUBSTR(x.i,incr+40,4)) /* Storage isolation    */
       lpi   = c2d(SUBSTR(x.i,incr+44,4)) / 100 /* Local PI       */
       spi   = c2d(SUBSTR(x.i,incr+48,4)) / 100 /* Sysplex PI     */
       sdata = c2d(SUBSTR(x.i,incr+52,4)) /* Offset to server sec.*/
       slen  = c2d(SUBSTR(x.i,incr+56,2)) /* Length of server sec.*/
       snum  = c2d(SUBSTR(x.i,incr+58,2)) /* Number of server ent.*/
/*-- -------------------------------------------------------------*/
/* Reformat goal value according to the goal type               */
/*-------------------------------------------------------------*/
        SELECT
          when gt=0 then gvvv='n/a'
          when gt=1 then gvvv=gval/1000
          when gt=2 then gvvv=gval/1000
          otherwise gvvv=gval
        END

rec99 = left(Date('N',smfdate,'J'),11) left(smftime,9),
        left(sysid,4)  left(syslvl,8)  left(sysname,8),
        left(sclass,8) left(period,1)  left(lpi,4),
```

```
          left(spi,4)    right(goal,15)  right(gvvv,3),
          left(imp,2)    right(pct,2)    left(dp,3),
          left(iodp,3)   right(mpli,2)   right(mplo,2),
          right(rua,4)   right(pspt,4)   right(psitar,4),
          right(sdata,4) right(slen,2)   right(snum,2)

   PUSH rec99
     "EXECIO 1 DISKW OUT99"

        end
      end
    end
  end
exit
```

It is strongly recommended that this report be used in conjunction with the RMF postprocessor service class reports, which allow us to look further into those workloads that are not performing as expected. These reports provide more detailed information about specific service classes. The RMF service class period report is created using the SYSRPTS(WLMGL(SCPER)) control card with the RMF postprocessor. Another way to gain a quick glance at service class performance and various types of resource delays is by using the RMF postprocessor overview record control statements. In the example below, the service class TSO period 1 is being examined. One would need to add control cards to specify other periods (ie second and third period if applicable and identical control cards for any other service classes one wants to report on).

Example:

```
//RMFSTEP1 EXEC PGM=ERBRMFPP,REGION=ØM
//MFPINPUT DD DISP=SHR,DSN=SMF.SORTED,DATASET
//MFPMSGDS DD SYSOUT=*
//*******************************************************************
//*GOAL MODE INDICATORS for TSO class - PI, USING AND DELAY SAMPLES *
//*******************************************************************
//SYSIN DD *
SYSOUT(O)
NOSUMMARY
OVERVIEW(REPORT)
DATE(MMDDYYYY,MMDDYYYY)
ETOD(Ø8ØØ,15ØØ)
OVW(PI(PI(S.TSO.1)),NOSYSTEMS)            /* PERFORMANCE INDEX,
SC.PERIOD 1 */
```

```
OVW(ENDEDTRX(TRANSTOT(S.TSO.1)),NOSYSTEMS)/* ENDED TRANSACTIONS */
OVW(VELOCITY(EXVEL(S.TSO.1)),NOSYSTEMS)   /* ACTUAL VELOCITY */
OVW(CPUUSING(CPUUSGP(S.TSO.1)),NOSYSTEMS) /* CPU USING % */
OVW(CPUDELAY(CPUDLYP(S.TSO.1)),NOSYSTEMS) /* CPU DELAY % */
OVW(IOUSING(IOUSGP(S.TSO.1)),NOSYSTEMS)   /* I/O USING % */
OVW(IODELAY(IODLYP(S.TSO.1)),NOSYSTEMS)   /* I/O DELAY % */
OVW(APPLPCT(APPLPER(S.TSO.1)),NOSYSTEMS)  /* APPLICATION % */
OVW(UNKNOWN(UNKP(S.TSO.1)),NOSYSTEMS)     /* UNKNOWN STATE % */
OVW(IDLE(IDLEP(S.TSO.1)),NOSYSTEMS)       /* IDLE STATE % */
OVW(SWPDELAY(SWINP(S.TSO.1)),NOSYSTEMS)   /* SWAP IN DELAY % */
OVW(MPLDELAY(MPLP(S.TSO.1)),NOSYSTEMS)    /* MPL DELAY % */
OVW(CAPDELAY(CAPP(S.TSO.1)),NOSYSTEMS)    /* CAPPING DELAY % */
OVW(DASDDISC(DISC(S.TSO.1)),NOSYSTEMS)    /* DASD DISCONNECT TIME */
OVW(DASDIOSQ(IOSQ(S.TSO.1)),NOSYSTEMS)    /* DASD IOSQ TIME */ /*
/*
```

Mile Pekic
Systems Programmer (Serbia and Montenegro)

Why not share your expertise and earn money at the same time? *MVS Update* is looking for macros, program code, JCL, REXX EXECs, etc, that experienced MVS users have written to make their life, or the lives of their users, easier.

We will publish it (after vetting by our expert panel) and send you a cheque when the article is published. Articles can be of any length and can be sent to Trevor Eddolls at any of the addresses shown on page 2. Alternatively, they can be e-mailed to trevore@xephon.com.

A free copy of our *Notes for contributors* is available from our Web site at www.xephon.com/nfc.

In addition to *MVS Update*, the Xephon family of *Update* publications includes *CICS Update*, *MQ Update*, *DB2 Update*, *TCP/SNA Update*, *RACF Update*, and *AIX Update*.

Comparing two files

The following program was written to check whether two files have identical contents. The files being compared can be KSDS, RRDS, ESDS, or non-VSAM. The files have DDnames INFILE1 and INFILE2. Each record is compared with the same record of the other file. If one of the records is shorter than the other, the shorter record is padded with the character indicated by variable PADCHAR within the program (space by default), and the rules of the CLCL (compare logical long) instruction apply. This is useful, for example, if you want to compare an ordinary 80-byte sequential file, but where only the first 50 bytes are meaningful and the remaining are spaces, with a 50-byte VSAM file.

If an unequal record is found, the program sends out a message to SYSPRINT, indicating that record number, and continues the comparison. If a predefined number of unequal records is attained (as defined by variable DIFLIMIT), the program terminates. Otherwise, the program continues until it reaches the end of both files. If one of the files ends first, the program continues to read the other up to its end, but without performing any more comparisons, in order to find out how many records each file has. In the end, it sends a message with that information.

You can also compare just part of each record. For example, you have code in the first file which is 10 bytes long and occurs at offset 23, and you want to know if it matches the second file, where that code occurs at offset 175. For this, pass a parameter to the program, with two pairs of length-offset values, with each value separated by commas. The first pair relates to INFILE1 and the second pair to INFILE2:

```
//STEP1  EXEC PGM=VCOMPARE,PARM='10,23,10,175'
//INFILE1  DD DISP=SHR,DSN=first.file
//INFILE2  DD DISP=SHR,DSN=second.file
//SYSPRINT DD SYSOUT=*
```

This way, only the specified bytes are compared, and the rest of each record is ignored.

If no parameter is given, the entire records are compared, as initially explained. Parameters are positional, and can be partially omitted. For example, if you just want to compare the first 25 bytes of each record (that means, with the default offset zero), you can code either of these:

```
//EXEC PGM=VCOMPARE,PARM='25,Ø,25,Ø'
//EXEC PGM=VCOMPARE,PARM='25,,25'
```

This also is applicable if, as in the first example, you just want to compare the first 50 bytes of the sequential file against the entire 50-byte VSAM. Since this last value is the default, because it is the VSAM record length, you can omit it, and just specify the length for the first file:

```
//EXEC PGM=VCOMPARE,PARM='5Ø'
```

which, in this particular case, would be identical to:

```
//EXEC PGM=VCOMPARE,PARM='5Ø,Ø,5Ø,Ø'
//EXEC PGM=VCOMPARE,PARM='5Ø,,5Ø'
```

This way, the remaining 30 bytes of the seqential file would be ignored, and only the first 50 bytes would be considered.

VCOMPARE SOURCE CODE

```
*=================================================================*
*                                                                 *
* VCOMPARE - Compare two files. The files can be VSAM, sequential,  *
* or both, with fixed or variable length.                         *
* Files are assigned to DDnames INFILE1 and INFILE2.              *
* Records are compared in a parallel fashion. If two records do not  *
* have the same length, the smaller is considered to be padded with  *
* the character stored in variable PADCHAR (space by default).    *
* If inequal records are found, the program prints a message with   *
* the record number and increments a counter. If a certain number of *
* unequal records is attained, the program terminates.            *
* That number is set in variable DIFLIMIT.                        *
*                                                                 *
* The program also states how many records each file has. If a file  *
* ends sooner than the other, the program continues to read the     *
* longer file until it ends, to determine the number of records, but *
* no more comparisons take place.                                 *
*                                                                 *
* Records can be compared totally or in part. For partial comparison *
* specify a length-offset pair for each file, with each value     *
```

```
* separated by commas. Values are positional. If a value is not      *
* specified, the defaults are assumed (the full length of each       *
* record from offset zero). The first pair of values concerns file1  *
* and the second pair concerns file2.                                 *
*                                                                     *
*=====================================================================*
&PROGRAM SETC  'VCOMPARE'
&PROGRAM AMODE 31
&PROGRAM RMODE 24
&PROGRAM CSECT
         SAVE  (14,12)
         LR    R12,R15
         USING &PROGRAM,R12
         ST    R13,SAVEA+4
         LA    R11,SAVEA
         ST    R11,8(R13)
         LR    R13,R11
         B     GETPARMS
         DC    CL16' &PROGRAM 2.1'
         DC    CL8'&SYSDATE'
*
*=====================================================================*
* Separate input parameter into its components, convert them to
* binary form, and store them in fields PARM1 thru PARM4. If any of
* the parms does not exist, those fields will remain with low-values.
* For each parm specified, set the corresponding flag field to 1, in
* order to allow CLI comparisons later on.
*=====================================================================*
*
GETPARMS DS    ØH
         LR    R2,R1               Copy parameter pointer to R2.
         L     R2,Ø(Ø,R2)          Load parm address
         LH    R3,Ø(R2)            Load parm length in R3
         LTR   R3,R3               Any parm entered?
         BZ    OPENPRT             No
*
         LR    R6,R2
         AR    R6,R3               R6: point after end of parms
         LA    R6,2(Ø,R6)          Skip 2 bytes of parmlength
         LA    R2,2(Ø,R2)          Skip 2 bytes of parmlength
         LR    R4,R2               R4: Current char to ckeck
         LA    R11,PARM1           Area to keep parms in binary form
         XR    R9,R9               Clear length counter
*
LOOPARMS EQU   *
         CR    R4,R6               End of all parms?
         BNL   CONVERT             Yes, go convert the last one
         CLI   Ø(R4),C','          Comma found?
         BE    CONVERT             Yes, go convert parm
         LA    R9,1(Ø,R9)          Increment index (char counter)
```

```
         LA    R4,1(Ø,R4)          Increment pointer (current char)
         B     LOOPARMS            And continue
*
CONVERT  EQU   *
         LTR   R9,R9               Any chars in current parm?
         BZ    CONVERT2            No, skip pack and cvb instructions
         S     R9,=F'1'            Sub one for ex
         EX    R9,PACKEX           Execute pack
         LA    R9,1(Ø,R9)          Increment again
         CVB   R7,PARMPACK         Convert to binary into R7
         ST    R7,Ø(R11)           And store it in R11 (Parm1 to 4)
*
CONVERT2 EQU   *
         CR    R4,R6               End of all parms?
         BNL   SETFLAG1            Yes, move ahead
         AR    R2,R9               Add length to base pointer
         LA    R2,1(Ø,R2)          And skip comma
         XR    R9,R9               Reset length
         LR    R4,R2               R4: Current char
         LA    R11,4(Ø,R11)        Point next binary parm storarea
         B     LOOPARMS
*
SETFLAG1 CLC   PARM1,=F'Ø'         Parm1 specified?
         BE    SETFLAG2            No, try next
         MVI   P1FLAG,C'1'         Yes, set flag to 1
SETFLAG2 CLC   PARM2,=F'Ø'         Same for others
         BE    SETFLAG3
         MVI   P2FLAG,C'1'
SETFLAG3 CLC   PARM3,=F'Ø'
         BE    SETFLAG4
         MVI   P3FLAG,C'1'
SETFLAG4 CLC   PARM4,=F'Ø'
         BE    OPENPRT
         MVI   P4FLAG,C'1'
*
*================================================================*
* Check what kind of files we have and try to open them.
* First attempt is to open as VSAM. If Error, assume non-VSAM file.
* If VSAM, test ACB for ESDS. If ESDS, modify RPL accordingly.
*================================================================*
*
OPENPRT  DS    ØF                  Open sysprint
         OPEN  (SYSPRINT,OUTPUT)   for displaying messages
*
OPENACB1 EQU   *                   Open ACB for VSAM file
         OPEN  INFILEA1            If error, go open DCB for seq file
         LTR   R15,R15
         BNZ   OPENDCB1
         TESTCB ACB=INFILEA1,                                      X
               ATRB=ESDS           Check if VSAM ESDS
```

```
         BNE   OPENACB2           No, go open second file
*
ESDSFIL1 EQU   *
         MODCB RPL=INFILER1,                                          X
               OPTCD=ADR          Modify RPL for ESDS
         B     OPENACB2
*
OPENDCB1 EQU   *                  Open DCB (sequential file)
         OPEN  (INFILED1,INPUT)
         LTR   R15,R15
         BNZ   ERRMSG2
*
         MVI   FILETYP1,C'S'      Set flag sequential type (nonVSAM)
         LA    R2,INFILED1        Address IHADCB of input file1
         USING IHADCB,R2
         TM    DCBRECFM,DCBBIT1   Is recfm V or U (B'x1xxxxxx)
         BNO   OPENACB2           No, jump ahead
         MVI   FILEVAR1,C'U'      set recfm undefined
         TM    DCBRECFM,DCBBITØ   Is recfm U (B'11xxxxxx)
         BO    OPENACB2           Yes, jump ahead
         MVI   FILEVAR1,C'V'      set recfm variable
*
*
OPENACB2 EQU   *                  Open ACB for VSAM file
         OPEN  INFILEA2           If error, go open DCB for seq file
         LTR   R15,R15
         BNZ   OPENDCB2
         TESTCB ACB=INFILEA2,                                         X
               ATRB=ESDS          Check if VSAM ESDS
         BNE   READFILS           No, jump ahead
*
ESDSFIL2 EQU   *
         MODCB RPL=INFILER2,                                          X
               OPTCD=ADR          Modify RPL for ESDS
         B     READFILS
*
OPENDCB2 EQU   *                  Open DCB (sequential file)
         OPEN  (INFILED2,INPUT)
         LTR   R15,R15
         BNZ   ERRMSG2
*
         MVI   FILETYP2,C'S'      Set flag sequential type (nonVSAM)
         LA    R11,INFILED2       Address IHADCB of input file2
         DROP  R2
         USING IHADCB,R11
         TM    DCBRECFM,DCBBIT1   Is recfm V or U (B'x1xxxxxx)
         BNO   READFILS           No, jump ahead
         MVI   FILEVAR2,C'U'      set recfm undefined
         TM    DCBRECFM,DCBBITØ   Is recfm U (B'11xxxxxx)
         BO    READFILS           Yes, jump ahead
```

```
         MVI   FILEVAR2,C'V'       set recfm variable
*
*=================================================================*
* Now enter a loop where we read a pair of records and compare them
* If they are equal, continue reading another pair.
* If they are different, print a message with the record number within
* the file and continue. If the maximum limit of different records
* is attained, terminate the program.
*=================================================================*
*
READFILS EQU   *
         XR    R8,R8               Record count for file1
         XR    R9,R9               Record count for file2
READ1    EQU   *
         CLI   ENDF1,C'F'          End of file1 already happened?
         BE    READ2               Yes, just read file2
         LA    R8,1(Ø,R8)          Increment file1 record counter
         CLI   FILETYP1,C'V'
         BNE   READSEQ1
*
READVSA1 EQU   *
         GET   RPL=INFILER1        Read VSAM file
         LTR   R15,R15             End of file?
         BNZ   ENDFILE1
         L     R4,VAREA1           Get address of data in R4.
         SHOWCB RPL=INFILER1,                                        X
               AREA=LRECL1,                                          X
               LENGTH=4,                                             X
               FIELDS=RECLEN
         L     R5,LRECL1           Get record length in R5
         B     READ2
*
READSEQ1 EQU   *
         DROP  R11
         USING IHADCB,R2
         GET   INFILED1            Read sequential (locate method)
         LR    R4,R1               copy address of data to R4.
         LH    R5,DCBLRECL         Load R5 with record length.
         CLI   FILEVAR1,C'V'       Is recfm variable?
         BNZ   READ2               No, jump ahead.
         LA    R4,4(Ø,R4)          Yes, skip 4 bytes of RDW
         SH    R5,=H'4'            And reduce record length.
*
READ2    EQU   *
         CLI   ENDF2,C'F'          End of file2 already happened?
         BE    READ1               Yes, just read file1
         LA    R9,1(Ø,R9)          Increment file2 record counter
         CLI   FILETYP2,C'V'
         BNE   READSEQ2
*
READVSA2 EQU   *
```

```
         GET   RPL=INFILER2        Read VSAM file
         LTR   R15,R15             End of file?
         BNZ   ENDFILE2
         L     R6,VAREA2           Get address of data in R6
         SHOWCB RPL=INFILER2,                                          X
               AREA=LRECL2,                                            X
               LENGTH=4,                                               X
               FIELDS=RECLEN
         L     R7,LRECL2           Get record length in R7
         B     COMPARE
*
READSEQ2 EQU   *
         DROP  R2
         USING IHADCB,R11
         GET   INFILED2            Read sequential (locate method)
         LR    R6,R1               copy address of data to R6.
         LH    R7,DCBLRECL         Load R7 with record length.
         CLI   FILEVAR2,C'V'       Is recfm variable?
         BNZ   COMPARE             No, jump ahead.
         LA    R6,4(Ø,R6)          Yes, skip 4 bytes of RDW
         SH    R7,=H'4'            And reduce record length.
*
COMPARE  EQU   *
         CLI   ENDF1,C'F'          If any of the files already ended,
         BE    READ2               no comparison is necessary.
         CLI   ENDF2,C'F'          Just continue to read the other
         BE    READ1               until it ends also.
*
ASKFLAG1 CLI   P1FLAG,C'Ø'         If parm1 (length) not zero,
         BE    ASKFLAG2            assume parm1 length for file1
         L     R5,PARM1
ASKFLAG2 CLI   P2FLAG,C'Ø'         If parm2 (offset) not zero,
         BE    ASKFLAG3            add it to the record pointer
         A     R4,PARM2
ASKFLAG3 CLI   P3FLAG,C'Ø'         Same for file 2 parms.
         BE    ASKFLAG4
         L     R7,PARM3
ASKFLAG4 CLI   P4FLAG,C'Ø'
         BE    ASKNOMOR
         A     R6,PARM4
ASKNOMOR ICM   R7,B'1ØØØ',PADCHAR  Insert padchar in R7
*
COMPLOOP EQU   *
         CLCL  R4,R6               Compare strings
         BZ    READ1               Strings are equal
         BL    DIFERENT            Strings are different
         BH    DIFERENT            Strings are different
         B     COMPLOOP            Equal so far, continue
*
DIFERENT EQU   *
         XR    RØ,RØ
```

```
         AH    RØ,DIFCOUNT         Increment different record counter
         AH    RØ,=H'1'
         STH   RØ,DIFCOUNT
         CH    R8,DIFLIMIT         Different limit attained?
         BE    EXIT2               Yes, exit
         LR    RØ,R8               Prepare different recnum
         BAL   R1Ø,UNPACK          for display
         MVC   DIFNUM,OUT1Ø
         PUT   SYSPRINT,DIFMSG     Send message
         B     READ1               And continue with next record
*
ENDFILE1 EQU   *
         S     R8,=F'1'
         LR    RØ,R8               Prepare number of records
         BAL   R1Ø,UNPACK          for display
         MVC   ENDNUM1,OUT1Ø
         PUT   SYSPRINT,ENDMSG1    Send message
         MVI   ENDF1,C'F'
         CLI   ENDF2,C'F'
         BE    EXITØ
         B     READ2
*
ENDFILE2 EQU   *
         S     R9,=F'1'
         LR    RØ,R9
         BAL   R1Ø,UNPACK
         MVC   ENDNUM2,OUT1Ø
         PUT   SYSPRINT,ENDMSG2
         MVI   ENDF2,C'F'
         CLI   ENDF1,C'F'
         BE    EXITØ
         B     READ1
*
*=====================================================================*
*          Close files and exit
*=====================================================================*
*
EXITØ    EQU   *
         CLC   DIFCOUNT,=H'Ø'
         BNE   EXIT1
         PUT   SYSPRINT,NODIFMSG
         B     EXIT1
*
EXIT2    EQU   *
         PUT   SYSPRINT,LIMITMSG
*
EXIT1    EQU   *
         CLOSE INFILED1
         CLOSE INFILED2
         CLOSE INFILEA1
         CLOSE INFILEA2
```

```
         CLOSE SYSPRINT
         L     R13,SAVEA+4
         LM    R14,R12,12(R13)
         XR    R15,R15
         BR    R14
*
*=================================================================*
*        Subroutines and work areas
*=================================================================*
*
UNPACK   EQU   *                  Binary to display:
         CVD   RØ,REGDECIM        Convert binary to packed decimal
         UNPK  OUT12,REGDECIM     and unpack
         BR    R1Ø                Return
*
ERRMSG1  EQU   *
         PUT   SYSPRINT,=CL8Ø'>>> Error opening input file INFILE1'
         B     EXIT1
ERRMSG2  EQU   *
         PUT   SYSPRINT,=CL8Ø'>>> Error opening input file INFILE2'
         B     EXIT1
*
INFILEA1 ACB   DDNAME=INFILE1       VSAM ACB
INFILER1 RPL   ACB=INFILEA1,        VSAM RPL                        X
               OPTCD=LOC,           Locate method                   X
               AREA=VAREA1,         Record buffer address           X
               ARG=CHAVE1           Only needed for rrds
*
INFILED1 DCB   DSORG=PS,MACRF=(GL), For sequential files            X
               EODAD=ENDFILE1,                                      X
               DDNAME=INFILE1
*
INFILEA2 ACB   DDNAME=INFILE2       VSAM ACB
INFILER2 RPL   ACB=INFILEA2,        VSAM RPL                        X
               OPTCD=LOC,           Locate method                   X
               AREA=VAREA2,         Record buffer address           X
               ARG=CHAVE2           Only needed for rrds
*
INFILED2 DCB   DSORG=PS,MACRF=(GL), For sequential files            X
               EODAD=ENDFILE2,                                      X
               DDNAME=INFILE2
*
SYSPRINT DCB   DSORG=PS,MACRF=(PM),                                 X
               LRECL=8Ø,                                            X
               DDNAME=SYSPRINT
*
         LTORG
SAVEA    DS    18F
VAREA1   DS    F                    Address of record buffer (VSAM)
CHAVE1   DS    F                    Record key (rrds - VSAM)
LRECL1   DS    F                    Record length (VSAM)
```

```
VAREA2   DS    F                    Address of record buffer (VSAM)
CHAVE2   DS    F                    Record key (rrds - VSAM)
LRECL2   DS    F                    Record length (VSAM)
FILETYP1 DC    C'V'                 Flag preset for VSAM
FILETYP2 DC    C'V'                 Flag preset for VSAM
FILEVAR1 DC    C'F'                 Flag preset to recfm F (if nonVSAM)
FILEVAR2 DC    C'F'                 Flag preset to recfm F (if nonVSAM)
ENDF1    DC    C' '                 Flag for end of file 1
ENDF2    DC    C' '                 Flag for end of file 2
PADCHAR  DC    C' '                 Fill char for different reclen
DIFLIMIT DC    H'2Ø'                Max different records
DIFCOUNT DC    H'Ø'                 Different record count area
*
PACKEX   PACK  PARMPACK,Ø(Ø,R2)     Pack from input parm to parmpack
PARMPACK DS    D                    Pack and convert to binary areas
PARM1    DC    F'Ø'                 for input parameters
PARM2    DC    F'Ø'
PARM3    DC    F'Ø'
PARM4    DC    F'Ø'
P1FLAG   DC    C'Ø'                 Flags are set to 1
P2FLAG   DC    C'Ø'                 if parms 1 to 4 have a value
P3FLAG   DC    C'Ø'                 other than the initial zero.
P4FLAG   DC    C'Ø'
*
         DS    ØD                   Convert to decimal and unpack
REGDECIM DS    CL9                  areas for output numbers
         DS    ØF
OUT12    DS    ØCL12
OUT1Ø    DS    CL1Ø
         DS    CL2
*
NODIFMSG DC    CL8Ø'** No differences found in compared records **'
LIMITMSG DC    CL8Ø'++ Max number of different records attained ++'
DIFMSG   DC    C'Differences found in record number :'
DIFNUM   DS    CL1Ø
         DC    CL4Ø' '
ENDMSG1  DC    C'Number of records of INFILE1 . . . :'
ENDNUM1  DS    CL1Ø
         DC    CL4Ø' '
ENDMSG2  DC    C'Number of records of INFILE2 . . . :'
ENDNUM2  DS    CL1Ø
         DC    CL4Ø' '
*
         DCBD  DSORG=PS             Ihadcb map (addressed by R2 for
         YREGS                      file1 and by R11 for file2)
         END
```

VSAM information giving advice for reorganization

There are several reasons why a VSAM dataset might need to be reorganized, including the following:

- Too many extents
- CA splits
- Too many CI splits
- Dataset is residing on the wrong disk.

Some people know everything about their VSAM datasets, but I'm not one of them. However, there are some things about a dataset that it's useful to know, for example information about its usage, such as how many records are inserted/deleted or updated, and how many records there are in the dataset. If a lot of records are inserted or deleted in one day, the dataset will need to be reorganized.

The space allocation of the dataset is also important. For example if you have a dataset with 15 extents of 500 cylinders you won't need to reorganize, whereas if the dataset has extents of four cylinders, reorganizing will improve performance.

Too many CI and CA splits spell disaster for the dataset. And too many extents consume storage in CICS, so you can improve performance by minimizing the extents.

If two heavy I/O-consuming datasets reside on one disk, you'll want to spread the I/O in order to get a better throughput.

In our company, some datasets are reorganized every day, others once a week, and the rest once a month. If any dataset needs to be reorganized more frequently, it will move to the job that runs more frequently (and conversely if it needs to be reorganized less frequently).

Once a week, the job presented in this article gives me the information I need. The coding is written in PL/I and will be

executed by JCL. If you're using another level of IDCAMS you'll need to change the record-layout of LC1, LC2, LC3, and LC4 to meet your requirements. At the end of the article I've reproduced an example of our IDCAMS output.

THE PL/I PROGRAM

```
 /****** VSAM INFO GIVING ADVICE FOR REORGANIZATION    =VSMINFO= ******/
 /*                                                                   */
 /*  PROGRAM        : VSMINFO                                         */
 /*                                                                   */
 /*  PURPOSE        : GET USEFUL INFORMATION NEEDED TO KNOW IF YOU    */
 /*                   NEED TO REORGANIZE VSAM DATASETS                */
 /*                                                                   */
 /*  INPUT          : IDCAMS : LISTOUTPUT FROM IDCAMS                 */
 /*                                                                   */
 /*  OUTPUT         : PRINTØ1 : LIST YOU WANT TO HAVE                 */
 /*                                                                   */
 /*  NOTES          : NONE                                            */
 /*                                                                   */
 /****** VSAM INFO GIVING ADVICE FOR REORGANIZATION    =VSMINFO= ******/

1VSMINFO:PROC OPTIONS(MAIN) REORDER;
 DCL IDCAMS FILE RECORD  SEQL INPUT ;
 DCL PRINTØ1 FILE RECORD  SEQL OUTPUT;
 ON ENDFILE (IDCAMS)     EOF      = '1'B;

 /* OUTPUTRECORD FROM IDCAMS                  */
 DCL REC                 CHAR(131) INIT (' ');

 /* FIND TYPE: INDEX /DATA AND DATASETNAME     */
 DCL 1 LC1               BASED(ADDR(REC)),
       2 DATA_INDEX      CHAR(Ø5),
       2 NVT1            CHAR(Ø2),
       2 FIND_NAME       CHAR(Ø9),
       2 NVT2            CHAR(Ø1),
       2 NAME            CHAR(44);

 /* USED TO DETECT RECORDS SPLITS AND EXTENTS  */
 DCL 1 LC2               BASED(ADDR(REC)),
       2 NVT1            CHAR(12),
       2 FIND_RECS       CHAR(Ø8),
       2 NVT2            CHAR(Ø2),
       2 RECS            CHAR(1Ø),
       2 NVT3            CHAR(Ø5),
       2 FIND_SPLITS     CHAR(Ø9),
       2 NVT4            CHAR(12),
       2 SPLITS          CHAR(Ø3),
```

```
      2 NVT5             CHAR(27),
      2 EXTENTS          CHAR(Ø2);

/* FIND SPACE PARAMETERS XX CYLS/TRKS         */
DCL 1 LC3                BASED(ADDR(REC)),
      2 NVT1             CHAR(Ø8),
      2 FIND_SPACE       CHAR(1Ø),
      2 NVT2             CHAR(Ø6),
      2 SPACE_TYPE       CHAR(Ø3),
      2 NVT3             CHAR(Ø1),
      2 SPACE            CHAR(Ø4);

/* FIND LRECL                                 */
DCL 1 LC4                BASED(ADDR(REC)),
      2 NVT1             CHAR(37),
      2 FIND_LRECL       CHAR(Ø8),
      2 NVT2             CHAR(11),
      2 LRECL            CHAR(Ø5);

/* OUTPUT LINES: HEADERS                      */
DCL 1 K1,
      2 ASA              CHAR(Ø1) INIT('1'),
      2 TEXT1            CHAR(1Ø) INIT(' * * * *  '),
      2 TEXT2            CHAR(29) INIT(' V S A M - I N F O R M A T I'),
      2 TEXT3            CHAR(28) INIT('O N   R E O R G A N I Z E'),
      2 TEXT4            CHAR(19) INIT('    D A T A S E T S'),
      2 TEXT5            CHAR(1Ø) INIT('  * * * * '),
      2 TEXT6            CHAR(Ø9) INIT(' '),
      2 TEXT7            CHAR(Ø7) INIT(' DATE:'),
      2 DATUM1           CHAR(Ø8) INIT(' '),
      2 TEXT8            CHAR(Ø4) INIT(' '),
      2 TEXT9            CHAR(Ø5) INIT('PAGE '),
      2 PAGE             PIC'ZZ9' INIT(Ø);

DCL 1 K2,
      2 ASA              CHAR(Ø1) INIT('-'),
      2 TEXT1            CHAR(44) INIT('DATASET-NAME'),
      2 NVT1             CHAR(Ø1) INIT(' '),
      2 TEXT2            CHAR(Ø7) INIT('CI - CA'),
      2 NVT2             CHAR(Ø1) INIT(' '),
      2 TEXT3            CHAR(Ø4) INIT('EXT.'),
      2 NVT3             CHAR(Ø2) INIT(' '),
      2 TEXT4            CHAR(22) INIT('----------------- REC'),
      2 TEXT5            CHAR(21) INIT('ORDS ----------------'),
      2 NVT4             CHAR(Ø2) INIT(' '),
      2 TEXT6            CHAR(14) INIT('---- SPACE --'),
      2 NVT5             CHAR(Ø2) INIT(' '),
      2 TEXT7            CHAR(11) INIT('-- LRECL --');

DCL 1 K3,
```

```
      2 ASA             CHAR(01) INIT(' '),
      2 NVT1            CHAR(45) INIT(' '),
      2 TEXT2           CHAR(07) INIT(' SPLITS'),
      2 NVT2            CHAR(07) INIT(' '),
      2 TEXT4           CHAR(22) INIT('     TOTAL   INSERTED '),
      2 TEXT5           CHAR(21) INIT('   DELETED    UPDATED'),
      2 NVT4            CHAR(16) INIT(' '),
      2 TEXT7           CHAR(13) INIT('  AVG.   MAX.');

/* OUTPUT LINES : DETAIL LINES                        */
DCL 1 D1,
      2 ASA             CHAR(01),
      2 NAME            CHAR(44),
      2 NVT1            CHAR(01),
      2 CI_SPLITS       CHAR(03),
      2 NVT2            CHAR(01),
      2 CA_SPLITS       CHAR(03),
      2 NVT3            CHAR(02),
      2 EXTENTS         CHAR(02),
      2 NVT4            CHAR(03),
      2 TOT_RECS        CHAR(10),
      2 NVT5            CHAR(01),
      2 INS_RECS        CHAR(10),
      2 NVT6            CHAR(01),
      2 DEL_RECS        CHAR(10),
      2 NVT7            CHAR(01),
      2 UPD_RECS        CHAR(10),
      2 NVT8            CHAR(02),
      2 SPACE_TYPE      CHAR(03),
      2 TEXT1           CHAR(01),
      2 SPACE_PRI       CHAR(04),
      2 TEXT2           CHAR(01),
      2 SPACE_SEC       CHAR(04),
      2 TEXT3           CHAR(01),
      2 NVT9            CHAR(01),
      2 AV_RECSIZE      CHAR(05),
      2 NVT10           CHAR(02),
      2 MAX_RECSIZE     CHAR(05);

DCL CATNAME             CHAR(08) BASED(ADDR(D1.NAME));

/* INITIALIZE DETAIL LINE                  */
D1 = ' ';
D1.TEXT1  = '(';
D1.TEXT2  = ',';
D1.TEXT3  = ')';


DCL (ADDR,DATE,SUBSTR)   BUILTIN;
```

```
/* HELP FIELDS                                */
DCL EOF                 BIT (Ø1) INIT ('Ø'B);
DCL I                   FIXED BIN (15);
DCL RECORD_COUNT        FIXED BIN (15) INIT(99);

/* START MAIN PROGRAM                         */
K1.DATUM1 = DATE;
K1.DATUM1 = SUBSTR(K1.DATUM1,5,2) || '-' ||
            SUBSTR(K1.DATUM1,3,2) || '-' ||
            SUBSTR(K1.DATUM1,1,2);

OPEN FILE (IDCAMS),
     FILE (PRINTØ1);
READ FILE (IDCAMS) INTO (REC);
DO   WHILE (¬EOF);

     /* SKIP USELESS LINES                     */
     DO   WHILE (LC1.FIND_NAME ¬= '--------' &
                 ¬EOF);
          REC = ' ';
          READ FILE (IDCAMS) INTO (REC);
          END;

     IF ¬EOF
     THEN DO;
          /* WE HAVE LINES WE COULD USE        */
          D1.NAME = LC1.NAME;
          IF CATNAME = 'CATALOG.'
          THEN DO;
               RECORD_COUNT = 99;
               REC = ' ';
               READ FILE (IDCAMS) INTO (REC);
               END;

          ELSE DO;

               /* RETRIEVE AND FILL THE DATA */
               DO   WHILE (LC4.FIND_LRECL ¬= 'AVGLRECL' &
                           ¬EOF);
                    REC = ' ';
                    READ FILE (IDCAMS) INTO (REC);
                    END;

               DO   I = 1 TO 5 WHILE (SUBSTR(LC4.LRECL,I,1) = '-');
                    SUBSTR(LC4.LRECL,I,1) = ' ';
                    END;
               D1.AV_RECSIZE = LC4.LRECL;

               IF D1.AV_RECSIZE = '    Ø'
```

```
THEN DO;
     D1.AV_RECSIZE  = ' ';
     D1.MAX_RECSIZE = ' ';
     END;
ELSE DO;

     DO   WHILE (LC4.FIND_LRECL ¬= 'MAXLRECL' &
                 ¬EOF);
          REC = ' ';
          READ FILE (IDCAMS) INTO (REC);
          END;

     DO   I = 1 TO 5
                WHILE (SUBSTR(LC4.LRECL,I,1) = '-');
          SUBSTR(LC4.LRECL,I,1) = ' ';
          END;
     D1.MAX_RECSIZE = LC4.LRECL;
     END;

DO   WHILE (LC2.FIND_SPLITS ¬= 'SPLITS-CI' &
            ¬EOF);
     REC = ' ';
     READ FILE (IDCAMS) INTO (REC);
     END;

DO   I = 1 TO 3 WHILE (SUBSTR(LC2.SPLITS,I,1) = '-');
           SUBSTR(LC2.SPLITS,I,1) = ' ';
           END;
D1.CI_SPLITS = LC2.SPLITS;

DO   I = 1 TO 1Ø WHILE (SUBSTR(LC2.RECS,I,1) = '-');
     SUBSTR(LC2.RECS,I,1) = ' ';
     END;
D1.TOT_RECS = LC2.RECS;

DO   WHILE (LC2.FIND_SPLITS ¬= 'SPLITS-CA' &
            ¬EOF);
     REC = ' ';
     READ FILE (IDCAMS) INTO (REC);
     END;

DO  I = 1 TO 1Ø WHILE (SUBSTR(LC2.RECS,I,1) = '-');
    SUBSTR(LC2.RECS,I,1) = ' ';
    END;
D1.DEL_RECS = LC2.RECS;

DO  I = 1 TO 3 WHILE (SUBSTR(LC2.SPLITS,I,1) = '-');
    SUBSTR(LC2.SPLITS,I,1) = ' ';
    END;
D1.CA_SPLITS = LC2.SPLITS;
```

```
DO  I = 1 TO 2 WHILE (SUBSTR(LC2.EXTENTS,I,1) = '-');
    SUBSTR(LC2.EXTENTS,I,1) = ' ';
    END;
D1.EXTENTS = LC2.EXTENTS;

DO   WHILE (LC2.FIND_RECS ¬= 'INSERTED' &
            ¬EOF);
     REC = ' ';
     READ FILE (IDCAMS) INTO (REC);
     END;

DO  I = 1 TO 1Ø WHILE (SUBSTR(LC2.RECS,I,1) = '-');
    SUBSTR(LC2.RECS,I,1) = ' ';
    END;
D1.INS_RECS = LC2.RECS;

DO   WHILE (LC2.FIND_RECS ¬= 'UPDATED-' &
            ¬EOF);
     REC = ' ';
     READ FILE (IDCAMS) INTO (REC);
     END;

DO  I = 1 TO 1Ø WHILE (SUBSTR(LC2.RECS,I,1) = '-');
    SUBSTR(LC2.RECS,I,1) = ' ';
    END;
D1.UPD_RECS = LC2.RECS;

DO   WHILE (LC3.FIND_SPACE ¬= 'SPACE-TYPE' &
            ¬EOF);
     REC = ' ';
     READ FILE (IDCAMS) INTO (REC);
     END;
IF LC3.SPACE_TYPE = 'CYL'
THEN D1.SPACE_TYPE = 'CYL';
ELSE D1.SPACE_TYPE = 'TRK';

DO   WHILE (LC3.FIND_SPACE ¬= 'SPACE-PRI-' &
            ¬EOF);
     REC = ' ';
     READ FILE (IDCAMS) INTO (REC);
     END;

DO  I = 1 TO 4 WHILE (SUBSTR(LC3.SPACE,I,1) = '-');
    SUBSTR(LC3.SPACE,I,1) = ' ';
    END;
D1.SPACE_PRI = LC3.SPACE;

DO   WHILE (LC3.FIND_SPACE ¬= 'SPACE-SEC-' &
            ¬EOF);
```

```
                    REC = ' ';
                    READ FILE (IDCAMS) INTO (REC);
                    END;

               DO  I = 1 TO 4 WHILE (SUBSTR(LC3.SPACE,I,1) = '-');
                   SUBSTR(LC3.SPACE,I,1) = ' ';
                   END;
               D1.SPACE_SEC = LC3.SPACE;

               /* PRINT ROUTINE                        */
               IF RECORD_COUNT > 6Ø
               THEN DO;
                    RECORD_COUNT  = 4;
                    K1.PAGE = K1.PAGE + 1;
                    WRITE FILE(PRINTØ1) FROM(K1);
                    WRITE FILE(PRINTØ1) FROM(K2);
                    WRITE FILE(PRINTØ1) FROM(K3);
                    D1.ASA = 'Ø';
                    END;

               WRITE FILE (PRINTØ1) FROM (D1);
               D1.ASA = ' ';
               RECORD_COUNT = RECORD_COUNT + 1;
               END;
          END;
     END;

 CLOSE FILE (IDCAMS),
       FILE (PRINTØ1);
 END VSMINFO;
```

THE JCL

```
//STREAM1   JOB CRC-57ØØ-Ø,'VSAM INFO',
//             CLASS=A,MSGCLASS=A
//*
//* PROJECT    : DISK UTILITIES
//* JOB        : LIST DATA TO KNOW WHICH VSAM DATASET HAS TO BE
//*              REORGANIZED
//*
//* STEPS      : NR |PROGRAM  |FUNCTION
//*              ---|---------|--------
//*              1Ø |IDCAMS   |LIST CATALOG(S)
//*              2Ø |VSMINFO  |PRINT VSAM INFO AND STATISTICS
//*
//*
//* SYSOUTSRT. : 1 : ON PAPER
//*
//* NOTES      : NONE
```

```
//*
//*
//* LIST CATALOGS IN WHICH THE VSAM DATASET ENTRIES IN EXIST
//STEP1Ø       EXEC PGM=IDCAMS
//SYSPRINT  DD DSN=IDCAMS.OUTPUT,DISP=(,CATLG),
//             DCB=(LRECL=131,BLKSIZE=279Ø7,RECFM=VBA),
//             UNIT=SYSDA,SPACE=(CYL,(1Ø,1Ø))
//SYSIN     DD *
    LISTCAT CATALOG (CATALOG.VSMICF1.VDISK1) -
        DATA INDEX ALL ;
    LISTCAT CATALOG (CATALOG.VSMICF1.VDISK2) -
        DATA INDEX ALL ;
/*
//*
//* REFORMAT THE OUTPUT OF IDCAMS TO GET THE DATA WE WANT TO SEE
//* IF A DATASET HAS TO BE REORGANISED
//STEP2Ø  EXEC PGM=VSMINFO
//IDCAMS    DD DSN=IDCAMS.OUTPUT,DISP=(OLD,DELETE)
//PRINTØ1   DD SYSOUT=1,DCB=(LRECL=137,RECFM=VA)
//SYSPRINT  DD SYSOUT=*
//*
```

SAMPLE IDCAMS OUTPUT

```
ØDATA ---------- VSAM.DATASET.ONE.DATA
      HISTORY
        DATASET-OWNER----(NULL)     CREATION--------2ØØ1.325
        RELEASE----------------2     EXPIRATION------ØØØØ.ØØØ
        ACCOUNT-INFO-----------------------------------(NULL)
      PROTECTION-PSWD----(NULL)     RACF----------------(NO)
      ASSOCIATIONS
        CLUSTER--VSAM.DATASET.ONE
      ATTRIBUTES
        KEYLEN-----------------11     AVGLRECL------------1ØØ
BUFSPACE----------1Ø24Ø     CISIZE--------------4Ø96

        RKP--------------------Ø     MAXLRECL------------512
EXCPEXIT----------(NULL)     CI/CA---------------18Ø

        SHROPTNS(3,3)      SPEED     UNIQUE           NOERASE
INDEXED        NWRITECHK     NOIMBED       NOREPLICATO

        UNORDERED          REUSE     NONSPANNED
      STATISTICS
        REC-TOTAL--------21Ø666     SPLITS-CI--------------Ø     EXCPS--
----------134338
        REC-DELETED------------Ø     SPLITS-CA-------------Ø
EXTENTS-------------1Ø2
REC-INSERTED----------Ø     FREESPACE-%CI--------25     SYSTEM-
```

```
1 * * * *   V S A M - I N F O R M A T IO N  R E O R G A N I Z E   D A T A S E T S  * * *  DATE: Ø7-Ø7-Ø2  PAGE  1
-DATASET-NAME                  CI - CA EXT. ----------------- RECORDS -------------  ---- SPACE --  -- LRECL --
                                SPLITS          TOTAL  INSERTED   DELETED   UPDATED                    AVG.   MAX.

ØVSAM.DATASET.ONE.DATA          Ø   Ø  Ø2      21Ø666         Ø         Ø         Ø    CYL(   2,   1)   1ØØ    512

 VSAM.DATASET.ONE.ONCE.DATA     Ø   Ø  79      143429         Ø         Ø         Ø    CYL(   2,   1)   1ØØ    512

 VSAM.DATASET.ONE.ONCE.INDEX    Ø   Ø   4          81         Ø         Ø         Ø    TRK(   1,   1)

 VSAM.DATASET.ONE.INDEX         Ø   Ø   5         1Ø4         Ø         Ø         Ø    TRK(   1,   1)

 VSAM.DATASET.TWO.DATA          Ø   Ø  36      244191         Ø         Ø         Ø    CYL(   2,   1)    9Ø    512

 VSAM.DATASET.TWO.INDEX         Ø   Ø   2          38         Ø         Ø         Ø    TRK(   1,   1)

 VSAM.DATASET.THREE.DATA      163   1   1         475       44Ø         1     273917    CYL(   2,   1)  13Ø7   13Ø7

 VSAM.DATASET.THREE.INDEX       1   Ø   1           3         Ø         Ø        22Ø  TRK(   1,   1)

 VSAM.DATASET.FOUR.DATA         Ø   Ø   1         476         Ø         1        23Ø  CYL(   1,   1)    16     16

 VSAM.DATASET.FOUR.INDEX        Ø   Ø   1           1         Ø         Ø          1  TRK(   1,   1)
```

Figure 1: Example job output

```
TIMESTAMP:
       REC-UPDATED-----------Ø     FREESPACE-%CA--------2Ø
X'B6C53CØ12AA92AØØ'
       REC-RETRIEVED----1685664     FREESPC--------15659ØØ8
     ALLOCATION
       SPACE-TYPE------CYLINDER     HI-A-RBA--------7593984Ø
       SPACE-PRI-------------2      HI-U-RBA--------7593984Ø
       SPACE-SEC-------------1
     VOLUME
VOLSER-----------DATA62     PHYREC-SIZE--------4Ø96     HI-A-RBA-------
-7593984Ø     EXTENT-NUMBER--------1Ø2

       DEVTYPE------X'3Ø1Ø2ØØF'     PHYRECS/TRK----------12     HI-U-
RBA--------7593984Ø     EXTENT-TYPE--------X'ØØ'

       VOLFLAG------------PRIME     TRACKS/CA-----------15
       EXTENTS:
       LOW-CCHH----X'Ø251ØØØØ'     LOW-RBA---------------Ø     TRACKS--
-------------3Ø

       HIGH-CCHH----X'Ø252ØØØE'     HIGH-RBA--------1474559
```

Sample job output is shown in Figure 1.

Teun Post
(The Netherlands)

Universal procedure for compiling and binding Enterprise PL/I programs

PROBLEM

We have more than 3,000 PL/I production programs in our MVS environment supporting our batch and online environment. Most of them work with DB2, but there are also old programs that work with sequential and VSAM datasets only. We planned to migrate from OS/390 V2R5 to z/OS 1.3 in one go. IBM didn't encourage users to leap eight steps at once, but we had some scheduling problems and were forced to migrate directly. To minimize the potential risks, we decided to migrate our production applications starting with source code.

Our previous experience in extensive manual compiling and linking was bad because this process is time-consuming and programmers were obliged to make load modules twice (for the current and for the new production) when any changes were made to the source code. Additionally, application departments developed new IT services at that time, so management requested automatic migration from the systems programmer group.

SOLUTION

We created a unique procedure for compiling and linking all program types. This procedure is functionally equivalent to four standard IBM procedures for compiling and binding:

- Batch programs without DB2
- CICS programs without DB2
- Batch programs with DB2
- CICS programs with DB2.

The Enterprise PL/I compiler, IBMZPLI, has integrated CICS and DB2 preprocessors, and we used this advantage for unification. This means that we can eliminate the preprocessor steps that were still present in the sample IBM procedures. On the other hand, we have to dynamically specify the appropriate compiler options and SYSLIN concatenation for the binder. In the past, when we had four different procedures, we specified a suitable compiler parameter in the PARM parameters of the preprocessor (PPLI) and compiler (PLI) step in each procedure. Users could change and add some options using the PLI.PARM parameter in the procedure call. Similarly, bind concatenation was customized in each procedure, and a user could influence it by the standard methods of replacing DD statements from a procedure call.

In the universal procedure, the preprocessor step doesn't exist and the compiler step is common for all types of program. This means that we can pass only compiler options that can be applied regardless of the program type. Options that are

dependent of the program type will be passed by using the %PROCESS directive. The purpose of the %PROCESS (synonym is *PROCESS) directive is to override compiler options. We can change source code by adding customized %PROCESS directives at the beginning of each member. It is the simplest method, but we don't want any source code changes.

Our solution was to concatenate dynamically-generated %PROCESS directives in front of the source code. This task is done by a REXX procedure that is the first step of the universal procedure for compile and bind.

The REXX procedure, named PLIPREP, also makes the customized concatenation for the BIND step.

DESCRIPTION OF PLIPREP

PLIPREP works without parameters in most cases. It scans the source code that the user specifies under the file name SOURCE, looking for CICS or DB2 statements. When the procedure recognizes the program type it generates suitable %PROCESS statements and writes it to a temporary dataset with the file name PROCESS. This temporary file will be the first dataset in the concatenation under the PLI.SYSIN DD statement. The procedure generates the following compiler options for all types of program:

```
*PROCESS MACRO INCLUDE OBJECT DEFAULT(DUMMY(UNALIGNED),UNALIGNED);
```

The first three options are necessary, option DEFAULT(DUMMY(UNALIGNED),UNALIGNED) is convenient for our installation because the default value is changed in the Ent PL/I compiler.

The other options are generated according to the scanning results and look like:

1 Program is batch without DB2:

```
*PROCESS MACRO INCLUDE OBJECT DEFAULT(DUMMY(UNALIGNED),UNALIGNED);
```

2 Program is batch with DB2:

```
*PROCESS MACRO INCLUDE OBJECT DEFAULT(DUMMY(UNALIGNED),UNALIGNED)
```

```
PP( SQL('HOST(PLI) GRAPHICS STDSQL(NO) CONNECT(2) TWOPASS'));
```

3 Program is CICS without DB2:

```
*PROCESS MACRO INCLUDE OBJECT DEFAULT(DUMMY(UNALIGNED),UNALIGNED)
 SYSTEM(CICS) PP( CICS(SP));
```

4 Program is CICS without DB2:

```
*PROCESS MACRO INCLUDE OBJECT DEFAULT(DUMMY(UNALIGNED),UNALIGNED)
 SYSTEM(CICS)
 PP( CICS(SP) SQL('HOST(PLI) GRAPHICS STDSQL(NO) CONNECT(2)
                                                  TWOPASS'));
```

Another function of the PLIPREP procedure is to generate appropriate BIND.SYSLIN DD concatenations. The procedure reads files with DD names PREP.CICSINC and PLIP.DB2INC that contain linkage editor instructions. CICSINC has instructions that the CICS manual recommends should be used in BIND.SYSLIN DD statement; DB2INC is similar. In the current release of CICS there is only one INCLUDE of DFHELII and one INCLUDE of DSNCLI in DB2. In future releases, it may be possible to have more than one instruction for the linkage editor. This is the reason behind putting them in the external datasets instead of generating simple statements directly. PLIPREP forms a temporary dataset with the file name BINDPARM and writes the required linkage editor instructions, depending of the program type. This dataset will be concatenated at the beginning of the BIND.SYSLIN DD statement.

In the process of writing modular applications, we create programs that call DB2 subroutines but they contain no EXEC SQL statements at all. In these cases, the PLIPREP procedure cannot recognize programs as a DB2 type – they must be compiled and linked as a DB2 program. A similar situation can arise with a CICS-type program if the installation extensively uses INCLUDE statements.

This problem will be solved in the one of the following ways.

- At our installation we established a convention similar to the IBM convention for recognizing a REXX procedure. A program that must be DB2 and/or CICS but has no EXEC SQL and/

or EXEC CICS statements must have a comment at the beginning with the keyword ENVIRONMENT followed by a parameter with the values CICS, DB2, or CICSDB2. Procedure PLIPREP scans the program, identifies the ENVIRONMENT keyword, checks its parameters, and generates appropriate customized output.

- The other solution is to pass a parameter to procedure PLIPREP. If you think that a source code change is not a good idea then you have to specify the ENV parameter of the procedure with the values identical to the ENVIRONMENT parameter.

CONCLUSION

In the process of migration we generated two types of job:

- PLIOBJ calls for compiling PL/I subroutines
- PLILOAD calls for compiling and binding PL/I programs.

By implementing this method we established our production environment in less then three hours. The only additional task was binding DB2 programs.

After a smooth migration, we continue to use the universal procedures for both the production and the test environments. Users could change compiler options through the PLIP parameter or change bind options using the BINDP parameter. The default compiler options and others can be influenced by using PARM of IBMZPLI. Options that we set by the %PROCESS directive remain unchanged thanks to IBM priorities.

PLIPREP REXX PROCEDURE

```
/******************************** REXX ***********************************/
/* Procedure generates: %PROCESS directive depending of PL/I program  */
/* type:                                                              */
/* a) Batch program                                                   */
/* b) Batch program with DB2                                          */
/* c) CICS program                                                    */
/* d) CICS program with DB2                                           */
```

```
/* -  concatenation of BIND.SYSLIN DD                                 */
/*                                                                    */
/* PARM: (optional)                                                   */
/*    CICS    - explicit specification of CICS environment            */
/*    DB2     - explicit specification of DB2 environment             */
/*    CICSDB2 - explicit specification of CICS and DB2 environment    */
/* Dataset:                                                           */
/*   SOURCE   - PL/I Source                                           */
/*   CICSINC  - file with required INCLUDE modules for CICS           */
/*   DB2INC   - file with required INCLUDE modules for DB2            */
/*   PROCESS  - generated %PROCESS directive                          */
/*   BINDPARM - generated control statements for BIND step            */
/********************************************************************/
/*  Trace ?R */

 ARG parm

rrc=Ø
"EXECIO Ø DISKR SOURCE (OPEN)"
 If RC <> Ø
 Then Do
      Say '>>> FILE SOURCE OPEN ERROR !!!'
      EXIT 12
      End

"EXECIO * DISKR  SOURCE (STEM Records.)"
"EXECIO Ø DISKR  SOURCE (FINIS)"

/*--Check whether member is suitable for inserting %PROCESS directive-*/
IndPROC = Ø
Do i = 1 To Records.Ø
   if index(Records.i,' PROC') > Ø | index(Records.i,':PROC') > Ø
   Then Do
        IndPROC = 1
        Leave
        End
End
If IndPROC = Ø
Then Do
     Say 'There is not PROC statements'
     Say 'This is not PL/I program'
     EXIT 12
     End

/*---------- Check whether program has CICS AND/OR DB2 statements ----*/
IndDB2 = Ø
IndCICS = Ø
if Parm = 'CICS' | Parm = 'CICSDB2'
Then IndCICS = 1
if Parm = 'DB2' | Parm = 'CICSDB2'
```

```
Then IndDB2 = 1
Do i = 1 To Records.Ø
   if index(Records.i,' EXEC ') > Ø
   Then Do
        if IndCICS = Ø & index(Records.i,' CICS ') > Ø
        Then IndCICS = 1
        if IndDB2 = Ø & index(Records.i,' SQL ') > Ø
        Then IndDB2 = 1
        End
   /*-- this is installation convention --*/
   if index(Records.i,' ENVIROMENT: ') > Ø
   Then Do
        if IndCICS = Ø & index(Records.i,'CICS') > Ø
        Then IndCICS = 1
        if IndDB2 = Ø & index(Records.i,'DB2') > Ø
        Then IndDB2 = 1
        End
   If IndDB2 = 1 & IndCICS = 1
   Then Leave
End

/*-------------- Generate %PROCESS directive ------------------------*/
"EXECIO Ø DISKW PROCESS (OPEN)"
 If RC <> Ø
 Then Do
      Say '>>> FILE SOURCE OPEN ERROR !!!'
      EXIT 12
      End

Process.Ø = Ø
Call ADD_PROCESS_Stmt

"EXECIO * DISKW  PROCESS   (STEM Process.)"
"EXECIO Ø DISKW  PROCESS   (FINIS)"

/*-------------- Generate BIND concatenation ------------------------*/
k = Ø
BINDParms.Ø = Ø
IF IndCICS = 1
Then CALL ADD_BIND_Parms CICSINC
IF IndDB2 = 1
Then CALL ADD_BIND_Parms DB2INC
BINDParms.Ø = k

"EXECIO * DISKW  BINDPARM  (STEM BINDParms.)"
"EXECIO Ø DISKW  BINDPARM  (FINIS)"
/*-------------- SAY Environment ----------------------------------*/
msg = ''
if IndCICS = 1
Then msg = msg || ' CICS '
```

```
Else msg = msg || ' BATCH'
if IndDB2 = 1
Then msg = msg || ' DB2'
Else msg = msg || '    '
SAY  ' Explicit PL/I Environment =' parm
SAY
SAY  ' Generated PL/I Environment =' msg
SAY
SAY  ' PL/I Process directive'
Do i = 1 To Process.Ø
   SAY Process.i
End
SAY
SAY  ' BIND Include'
Do i = 1 To BINDParms.Ø
   SAY BINDParms.i
End

Return rrc

/*-------------------------------------------------------------------*/
/* Open File                                                         */
/*-------------------------------------------------------------------*/
 OPEN_FILE:Procedure
 ARG File
 "EXECIO Ø DISKR "File" (OPEN)"
 If RC <> Ø
 Then Do
      Say '>>> FILE "File" OPEN ERROR !!!'
      EXIT 12
      End
 Return Ø

/*-------------------------------------------------------------------*/
/* Concatenation of parameters in %PROCESS directive                 */
/*-------------------------------------------------------------------*/
ADD_PROCESS_Stmt: Procedure EXPOSE Records. Process. IndCICS IndDB2

/*------------------ Get old %PROCESS directives --------------------*/
OldProcess = ''
IndProcess = Ø
Do i = 1 To Records.Ø
   parse upper var records.i PocessStmt parameters
   if processStmt = '%PROCESS'
   Then IndProcess = 1

   if IndProcess  = 1
   Then OldProcess = OldProcess || Records.i

   if IndProcess  = Ø & Index(Records.i,';') > Ø
```

```
   Then Leave

   if IndProcess  = 1 & Index(Records.i,';') > Ø
   Then IndProcess = Ø
End

/*================ Make new %PROCESS directive ====================*/
NewProcess = '%PROCESS'
LenNewProcess = 8
k = Ø

if index(OldProcess,' M ') = Ø
Then Call ADD_Options 'MACRO'
if index(OldProcess,'INC') = Ø
Then Call ADD_Options 'INCLUDE'
if index(OldProcess,'OBJ') = Ø
Then Call ADD_Options 'OBJECT'

/*-- default for Enterprise PL/I compiler is ALIGNED but it is ----*/
/*-- not appropriate in most cases                              ----*/

if index(OldProcess,'ALIGNED') = Ø
Then Call ADD_Options 'DEFAULT(DUMMY(UNALIGNED),UNALIGNED)'

/*-- This option  is required for CICS program --------------------*/
if IndCICS = 1
Then Call ADD_Options 'SYSTEM(CICS)'

/*-------- Preprocessor options for CICS & DB2 --------------------*/
if (IndDB2 = 1 | IndCICS = 1) & index(OldProcess,'PP') = Ø
Then Do
     Call ADD_Options 'PP('

     if IndCICS = 1
     Then Call ADD_Options 'CICS(SP)'
     if IndDB2 = 1
     Then Call ADD_Options 'SQL(''HOST(PLI) STDSQL(NO) CONNECT(2) ',
                           'NOGRAPHICS TWOPASS'')'
     Call ADD_Options ')'
     End
Call ADD_Options ';'

if LenNewProcess > Ø
Then Do
     k = k + 1
     Process.k = NewProcess
     End
Process.Ø = k
Return Ø
```

```
/*-------------------------------------------------------------*/
/* Options concatenation in %PROCESS directive                 */
/*-------------------------------------------------------------*/
 ADD_Options:Procedure Expose k OldProcess NewProcess LenNewProcess ,
                              Process.
 ARG options
 if index(OldProcess,options) = Ø
 Then Do
      LenOptions = Length(options)
      if LenNewProcess + LenOptions > 71
      Then Do
           k = k + 1
           Process.k = NewProcess
           NewProcess = ' '
           LenNewProcess = 1
           End
      NewProcess = NewProcess||' '||options
      LenNewProcess = LenNewProcess + LenOptions + 1
      End
 Return Ø

/*-------------------------------------------------------------*/
/* ADD of control statements for binder                        */
/*-------------------------------------------------------------*/
 ADD_BIND_Parms:Procedure Expose k BINDParms.
 ARG File

     "EXECIO Ø DISKR "File" (OPEN)"
     If RC <> Ø
     Then Do
          Say '>>> FILE "File" OPEN ERROR !!!'
          EXIT 12
          End
     "EXECIO * DISKR "File" (STEM Parms.)"
     "EXECIO Ø DISKR "File" (FINIS)"
     Do i = 1 To Parms.Ø
        k = K + 1
        BINDParms.k = Parms.i
     End
 Return Ø
```

PLILOAD PROCEDURE

```
//PLILOAD PROC LNGPRFX='IBMZ',LIBPRFX='CEE',
//        SYSLBLK=32ØØ,
//        INDEX='CICSTS22.CICS',       QUALIFIER(S) FOR CICS LIBRARIES
//        MACLIB='MACLIB',             PRIVATE MACRO/DSECT
//        NAME=,
//        PREFIX='TEST',
```

```
//        TYPE='BATCH',
//        ENV=,
//        PLIP='NGN',                  USER COMPILER OPTION
//        BINDP=                       USER BIND OPTION
//*
//*******************************************************************
//* IBM ENTERPRISE PL/I FOR Z/OS AND OS/390
//* VERSION 3 RELEASE 1 MODIFICATION 0
//*
//*  COMPILE AND BIND A PL/I PROGRAM
//*******************************************************************
//*-----------------------------------------------------------------
//*- EXECUTE REXX PLI PREPROCESOR
//*-----------------------------------------------------------------
//PLIPREP   EXEC PGM=IKJEFT01,DYNAMNBR=50,
//          PARM=('%PLIPREP &ENV')
//SYSPROC   DD  DSN=SYS1.CMDPROC,DISP=SHR
//SYSTSPRT  DD  SYSOUT=*
//SYSPRINT  DD  SYSOUT=*
//SOURCE    DD  DSN=&PREFIX..&TYPE..PLI(&NAME),DISP=SHR
//CICSINC   DD  DSN=&INDEX..SDFHC370(DFHEILID),DISP=SHR
//DB2INC    DD  DSN=DSN710.SDSNSAMP($CICS),DISP=SHR
//PROCESS   DD  DSN=&&PROCESS,DISP=(NEW,PASS),
//          UNIT=SYSDA,DCB=(RECFM=FB,LRECL=80,BLKSIZE=0),
//          SPACE=(TRK,(1))
//BINDPARM  DD  DSN=&&BINDPARM,DISP=(NEW,PASS),
//          UNIT=SYSDA,DCB=(RECFM=FB,LRECL=80,BLKSIZE=6160),
//          SPACE=(TRK,(1))
//SYSTSIN   DD  DUMMY
//*******************************************************************
//* COMPILE STEP
//*******************************************************************
//PLI      EXEC PGM=IBMZPLI,REGION=0M,
//          PARM=('S,OP,NEST,A,XREF,OBJ,INC,AG,A',
//          'OPT(2),M,OF,NUM,INSOURCE',&PLIP)
//STEPLIB  DD  DSN=&LNGPRFX..SIBMZCMP,DISP=SHR
//         DD  DSN=&LIBPRFX..SCEERUN,DISP=SHR
//         DD  DSN=&INDEX..SDFHLOAD,DISP=SHR
//SYSPRINT DD  SYSOUT=*
//SYSOUT   DD  SYSOUT=*
//SYSUT1   DD  DSN=&&SYSUT1,UNIT=SYSALLDA,
//             SPACE=(1024,(200,50),,CONTIG,ROUND),DCB=BLKSIZE=1024
//SYSLIB   DD  DSN=&PREFIX..&TYPE..PLI,DISP=SHR
//         DD  DSN=&PREFIX..&TYPE..&MACLIB,DISP=SHR
//         DD  DSN=DSN710.SRCLIB.DATA,DISP=SHR
//         DD  DSN=CICSTS22.CICS.SDFHPL1,DISP=SHR
//         DD  DSN=&INDEX..SDFHPL1,DISP=SHR
//         DD  DSN=&INDEX..SDFHMAC,DISP=SHR
//         DD  DSN=&INDEX..SDFHSAMP,DISP=SHR
//DBRMLIB  DD  DSN=&PREFIX..DBRMLIB.MACLIB(&NAME),DISP=SHR
```

```
//SYSIN    DD  DSN=&&PROCESS,DISP=(SHR,DELETE)
//         DD  DSN=&PREFIX..&TYPE..PLI(&NAME),DISP=SHR
//SYSLIN   DD  DSN=&SYSUID..&NAME..OBJ,DISP=(MOD,PASS),UNIT=SYSALLDA,
//             SPACE=(CYL,(1,1)),DCB=(LRECL=8Ø,BLKSIZE=&SYSLBLK)
//*******************************************************************
//* BIND STEP
//*******************************************************************
//BIND     EXEC PGM=IEWBLINK,COND=(8,LT,PLI),REGION=2Ø48K,
//             PARM=('LIST,XREF,MAP,COMPAT=PM3',&BINDP)
//SYSLIB   DD  DSN=&LIBPRFX..SCEELKED,DISP=SHR
//         DD  DSN=&INDEX..SDFHLOAD,DISP=SHR
//         DD  DSN=DSN71Ø.SDSNLOAD,DISP=SHR
//         DD  DSN=&PREFIX..&TYPE..OBJ,DISP=SHR
//         DD  DSN=TCPIP.SEZATCP,DISP=SHR
//SYSPRINT DD  SYSOUT=*
//SYSLIN   DD  DSN=&&BINDPARM,DISP=(OLD,DELETE)
//         DD  DSN=*.PLI.SYSLIN,DISP=(OLD,DELETE)
//         DD  DDNAME=SYSIN
//SYSLMOD  DD  DSN=&PREFIX..&TYPE..LOAD(&NAME),DISP=SHR
//SYSDEFSD DD  DUMMY
//SYSIN    DD  DUMMY
//*
```

PLIOBJ PROCEDURE

```
//PLIOBJ  PROC LNGPRFX='IBMZ',LIBPRFX='CEE',
//        INDEX='CICSTS22.CICS',       QUALIFIER(S) FOR CICS LIBRARIES
//        MACLIB='MACLIB',             PRIVATE MACRO/DSECT
//        NAME=,
//        PREFIX='TEST',
//        TYPE='BATCH',
//        ENV=,
//        PLIP='NGN'
//*
//*******************************************************************
//* IBM ENTERPRISE PL/I FOR Z/OS AND OS/39Ø
//* VERSION 3 RELEASE 1 MODIFICATION Ø
//*
//*  COMPILE A PL/I PROGRAM
//*******************************************************************
//*-----------------------------------------------------------------
//*- EXECUTE REXX PLI PREPROCESOR
//*-----------------------------------------------------------------
//PLIPREP   EXEC PGM=IKJEFTØ1,DYNAMNBR=5Ø,
//          PARM=('%PLIPREP &ENV')
//SYSPROC   DD  DSN=SYS1.CMDPROC,DISP=SHR
//SYSTSPRT  DD  SYSOUT=*
//SYSPRINT  DD  SYSOUT=*
//SOURCE    DD  DSN=&PREFIX..&TYPE..PLI(&NAME),DISP=SHR
```

```
//CICSINC   DD  DSN=&INDEX..SDFHC37Ø(DFHEILID),DISP=SHR
//DB2INC    DD  DSN=DSN71Ø.SDSNSAMP($CICS),DISP=SHR
//PROCESS   DD  DSN=&&PROCESS,DISP=(NEW,PASS),
//          UNIT=SYSDA,DCB=(RECFM=FB,LRECL=8Ø,BLKSIZE=Ø),
//          SPACE=(TRK,(1))
//BINDPARM  DD  DUMMY,DCB=BLKSIZE=8Ø
//SYSTSIN   DD  DUMMY
//*******************************************************************
//* COMPILE STEP
//*******************************************************************
//PLI       EXEC PGM=IBMZPLI,REGION=ØM,
//          PARM=('S,OP,NEST,A,XREF,OBJ,INC,AG,A',
//          'OPT(2),M,OF,NUM,INSOURCE',&PLIP)
//STEPLIB   DD  DSN=&LNGPRFX..SIBMZCMP,DISP=SHR
//          DD  DSN=&LIBPRFX..SCEERUN,DISP=SHR
//          DD  DSN=&INDEX..SDFHLOAD,DISP=SHR
//SYSPRINT  DD  SYSOUT=*
//SYSOUT    DD  SYSOUT=*
//SYSUT1    DD  DSN=&&SYSUT1,UNIT=SYSALLDA,
//              SPACE=(1Ø24,(2ØØ,5Ø),,CONTIG,ROUND),DCB=BLKSIZE=1Ø24
//SYSLIB    DD  DSN=&PREFIX..&TYPE..PLI,DISP=SHR
//          DD  DSN=&PREFIX..&TYPE..&MACLIB,DISP=SHR
//          DD  DSN=DSN71Ø.SRCLIB.DATA,DISP=SHR
//          DD  DSN=CICSTS22.CICS.SDFHPL1,DISP=SHR
//          DD  DSN=&INDEX..SDFHPL1,DISP=SHR
//          DD  DSN=&INDEX..SDFHMAC,DISP=SHR
//          DD  DSN=&INDEX..SDFHSAMP,DISP=SHR
//DBRMLIB   DD  DSN=&PREFIX..DBRMLIB.MACLIB(&NAME),DISP=SHR
//SYSIN     DD  DSN=&&PROCESS,DISP=(SHR,DELETE)
//          DD  DSN=&PREFIX..&TYPE..PLI(&NAME),DISP=SHR
//SYSLIN    DD  DSN=&PREFIX..&TYPE..OBJ(&NAME),DISP=SHR
```

SAMPLE JOB FOR PLILOAD AND PLIOBJ EXECUTION

```
//jobname JOB ...
//* programs are in common production libraries
//SUBROUT EXEC PLIOBJ,PREFIX=PROD,TYPE=CICS,NAME=subrnam1
//PROG1   EXEC PLILOAD,PREFIX=PROD,TYPE=BATCH,NAME=prognam1
//PROG2   EXEC PLILOAD,PREFIX=PROD,TYPE=CICS,NAME=prognam2,BINDP=RENT
//* programs are in common test libraries
//SUBROUT EXEC PLIOBJ,PREFIX=TEST,TYPE=BATCH,NAME=subrnam2,PLIP=GN
//PROG1   EXEC PLILOAD,PREFIX=TEST,TYPE=CICS,NAME=prognam3,PLIP=GN
//PROG2   EXEC PLILOAD,PREFIX=TEST,TYPE=BATCH,NAME=prognam4
//* programs are in user private libraries
//USER1   EXEC PLILOAD,PREFIX=username,TYPE=usertype,NAME=prognam5
```

Emina Spasic and Dragan Nikolic
Systems Programmers
Postal Savings Bank (Yugoslavia)

Extending the standard module design

In our last article (*Standard module structure and design*, *MVS Update*, Issue 201, June 2003) we examined a proposed structure for load modules. The structure that we proposed was simple and very easy to implement. Our experience has been that by utilizing this standardized structure we have realized productivity gains in the initial programming as well as in on-going maintenance activities. If you recall, the proposed module structure looked like Figure 1.

Each of the sections of the module was defined as a CSECT. We would like to take this basic structure and extend it.

At a high level, the executable code section of our module can really be broken down into three sections. We will have the actual portion of the program that is the executing code, and we will have a data portion of the program (which usually comprises

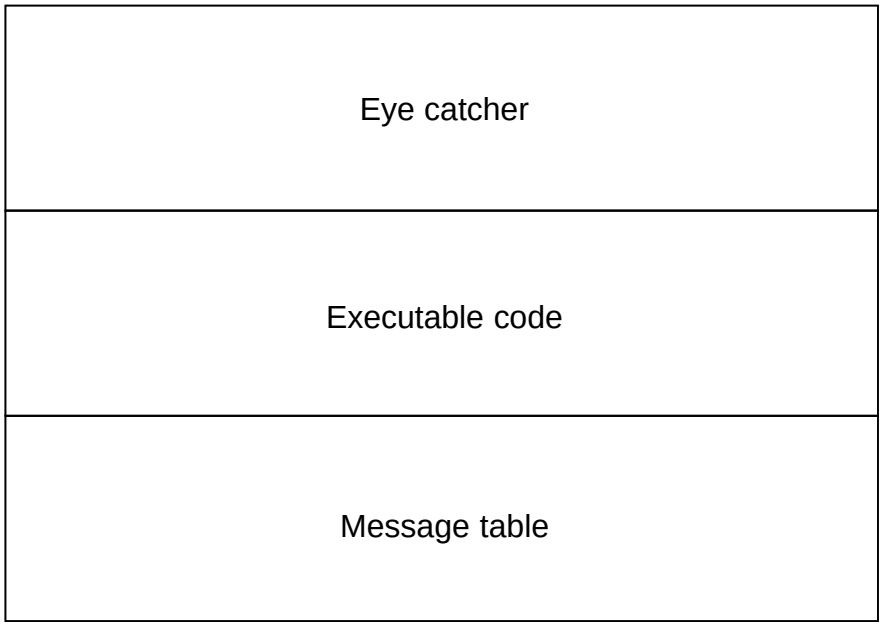

Figure 1: Proposed module structure

data elements that are static and do not change), and those storage locations that we will store information into. A counter would be a good example of this type of data element. Over time, we have made a conscious decision to code all of our Assembler programs as re-entrant. Not every program has a re-entrant requirement, but there is no harm or significant overhead in coding in this fashion, and again it facilitates the standardization objective. The macro processing capabilities of the Assembler provide an excellent vehicle for us to achieve both the standard structure as well as enjoy some level of productivity. We have four macros, which we will share, that begin to help us create and maintain our standard module structure. There is nothing unique about these macros. They provide standard entry and exit linkage. Bear in mind that the goal is standardization and productivity. The macros that we use are $EDTPRO, $EDTEPI, $EDTSTG, and $EDTEND. We will look at each of these individually.

$EDTPRO

The $EDTPRO is our prolog macro. We will use it at the beginning of the program to provide our base program linkage and to acquire a dynamic storage area that we can use for elements whose contents will change during program execution.

```
         MACRO
&LABEL   $EDTPRO &AM=31,&RM=ANY,&MODE=P
.*****************************************************************
.*                                                               *
.*       THIS MACRO WILL PROVIDE ENTRY LINKAGE AND OPTIONALLY    *
.*       MULTIPLE BASE REGISTERS.  TO USE THIS MACRO, YOU NEED TO*
.*       ALSO USE THE $EDTSTG MACRO.  THE $EDTSTG DEFINES THE SYMBOL *
.*       QLENGTH WHICH OCCURS IN THE CODE THAT &EDTPRO GENERATES.*
.*       IF YOU DO NOT CODE ANY OPERANDS, THEN REGISTER 12 WILL BE *
.*       USED AS THE BASE.  IF YOU CODE MULTIPLE SYMBOLS, THEN THEY *
.*       WILL BE USED AS THE BASE REGISTERS.                     *
.*                                                               *
.*       EXAMPLES:                                               *
.*                                                               *
.*             SECTNAME $EDTPRO          = REG 12 BASE           *
.*             SECTNAME $EDTPRO 5        = REG 5 BASE            *
.*             SECTNAME $EDTPRO R1Ø,R11  = REGS 1Ø AND 11 ARE BASES *
.*                                                               *
```

```
.*********************************************************************
.*       FOLLOWING GLOBAL VARIABLES ARE USED TO INCLUDE DSECTS WHEN   *
.*       NEEDED BY OTHER MACROS.                                      *
.*********************************************************************
         GBLC  &MY_CSECT
         GBLC  &PSA
         GBLC  &ASCB
         GBLC  &DCBD
         GBLC  &DCBE
.*********************************************************************
.*       SET UP SOME LOCAL VARIABLES.                                 *
.*********************************************************************
         LCLA  &AA,&AB,&AC
         LCLA  &L_SYSASM
         LCLA  &L_SYSIN_DSN
         LCLA  &L_SYSIN_MEMBER
.*********************************************************************
.*       INITIALIZE THE DSCET GLOBAL VARIABLES.                       *
.*********************************************************************
&MY_CSECT SETC '&LABEL'
&PSA      SETC  'NO'
&ASCB     SETC  'NO'
&DCBD     SETC  'NO'
&DCBE     SETC  'NO'
*
RØ        EQU   Ø
R1        EQU   1
R2        EQU   2
R3        EQU   3
R4        EQU   4
R5        EQU   5
R6        EQU   6
R7        EQU   7
R8        EQU   8
R9        EQU   9
R1Ø       EQU   1Ø
RA        EQU   1Ø
R11       EQU   11
RB        EQU   11
R12       EQU   12
RC        EQU   12
R13       EQU   13
RD        EQU   13
R14       EQU   14
RE        EQU   14
R15       EQU   15
RF        EQU   15
          SPACE 1
ARØ       EQU   Ø
AR1       EQU   1
```

```
AR2      EQU   2
AR3      EQU   3
AR4      EQU   4
AR5      EQU   5
AR6      EQU   6
AR7      EQU   7
AR8      EQU   8
AR9      EQU   9
AR1Ø     EQU   1Ø
ARA      EQU   1Ø
AR11     EQU   11
ARB      EQU   11
AR12     EQU   12
ARC      EQU   12
AR13     EQU   13
ARD      EQU   13
AR14     EQU   14
ARE      EQU   14
AR15     EQU   15
ARF      EQU   15
         SPACE 1
FPRØ     EQU   Ø
FPR2     EQU   2
FPR4     EQU   4
FPR6     EQU   6
         SPACE 1
         MNOTE *,'ASSEMBLER USED = &SYSASM'
         MNOTE *,'SYSIN DSN =      &SYSIN_DSN'
         MNOTE *,'SYSIN MEMBER   = &SYSIN_MEMBER'
.*************************************************************************
.*       INITIALIZE THE DSCET GLOBAL VARIABLES.                          *
.*************************************************************************
&LABEL   CSECT
&LABEL   AMODE &AM
&LABEL   RMODE &RM
         SPACE 1
         SYSSTATE ASCENV=&MODE         SET THE ENVIRONMENT
         SPACE 1
         BAKR  R14,Ø                   SAVE GPRS AND ARS ON THE STACK
         AIF   (N'&SYSLIST EQ Ø).USER12
         LAE   &SYSLIST(1),Ø(R15,Ø)    LOAD OUR BASE REG
         USING &LABEL,&SYSLIST(1)      LET THE ASSEMBLER KNOW
         AGO   .GNBASE
.USER12  ANOP
         MNOTE *,'NO BASE REG SPECIFIED, REGISTER 12 USED'
         LAE   R12,Ø(R15,Ø)            LOAD OUR BASE REG
         USING &LABEL,R12              LET THE ASSEMBLER KNOW
         AGO   .STGOB
.GNBASE  ANOP
         AIF   (N'&SYSLIST LE 1).STGOB
```

```
&AA      SETA  2
&AC      SETA  4Ø96
.GNBASE1 ANOP
         SPACE 1
         AIF   (&AA GT N'&SYSLIST).STGOB
&AB      SETA  &AA-1
         LR    &SYSLIST(&AA),&SYSLIST(&AB) GET INITIAL BASE
         A     &SYSLIST(&AA),$$$$4Ø96      ADJUST NEXT BASE
         USING &LABEL+&AC,&SYSLIST(&AA)    LET THE ASSEMBLER KNOW
&AA      SETA  &AA+1
&AC      SETA  &AC+4Ø96
         AGO   .GNBASE1
.STGOB   ANOP
         SPACE 1
         L     RØ,QLENGTH              GET THE DSECT LENGTH
         SPACE 1
         STORAGE OBTAIN,LENGTH=(RØ),LOC=(RES,ANY)
         SPACE 1
         LR    R15,R1                  GET @(OBTAINED AREA)
         L     R13,QDSECT              GET DISPLACEMENT INTO AREA
         LA    R13,Ø(R13,R15)          GET @(OBTAINED AREA)
         LR    RØ,R13                  SET REG Ø = REG 13
         L     R1,QLENGTH              GET THE LENGTH OF THE AREA
         XR    R15,R15                 CLEAR REG 5
         MVCL  RØ,R14                  INTIALIZE THE AREA
         MVC   4(4,R13),$$$$F1SA       INDICATE STACK USAGE
         USING DSECT,R13               INFORM ASSEMBLER OF BASE
.MEND    ANOP
         SPACE 1
         EREG  R1,R1                   RESTORE REGISTER 1
         MEND
```

If you look at the executable code that the macro generates, you will note that it provides the standard set-up and linkage required for a program. The DSECT label will reference the dynamic storage area that will contain modifiable variables. The length of this area is determined by the Assembler at assembly time and is assigned to the variable QLENGTH. We use this value to perform the STORAGE OBTAIN process. We have chosen to use register 13 as the base for the dynamic area. The EREG instruction is used to restore the contents of register 1. This is done for those programs that will inspect register 1 for a pointer to passed parameters.

If you scan the first portion of the macro you can see that we also provide the standard register equates as well as equates for the

access registers and the floating point registers. You will also notice that we have defined some global symbols. These are provided to help control the inclusion of other DSECTs that may be needed for the program assembly.

$EDTEPI

Now let's take a look at our epilogue code that is generated by the $EDTEPI macro.

```
         MACRO
         $EDTEPI
.*************************************************************************
.*
.*       THIS MACRO WILL PROVIDE EXIT LINKAGE. IT WILL FREE THE
.*       STORAGE AREA THAT WAS ACQUIRED BY THE $EDTPRO MACRO.  YOU
.*       CAN OPTIONALLY PASS IT A RETURN CODE VALUE.  THIS VALUE IS
.*       EITHER THE LABEL OF A FULL WORD IN STORAGE, OR IT IS A REG-
.*       ISTER. AS WITH THE $EDTPRO MACRO, YOU NEED TO USE THE $EDTSTG
.*       MACRO.  THE SYMBOL QLENGTH WHICH OCCURS IN THE CODE THAT IS
.*       GENERATED BY THIS MACRO IS DEFINED BY $EDTSTG
.*
.*       EXAMPLES:
.*
.*              $EDTEPI          = NO RETURN CODE SPECIFIED
.*              $EDTEPI (R5)     = RETURN CODE IS IN REG 5
.*              $EDTEPI RETCODE  = RETURN CODE IS IN THE FULLWORD AT
.*                                 RETCODE
.*
.*************************************************************************
         SPACE 1
.*************************************************************************
.*       DEFINE SOME LOCAL SYMBOLS
.*************************************************************************
         LCLC  &NO24
         LCLC  &NO31
.*************************************************************************
         AIF   (N'&SYSLIST EQ Ø).STGFRE
         SPACE 1
         AIF   ('&SYSLIST(1)'(1,1) EQ '(').REGRC
         L     R2,&SYSLIST(1)          GET RETURN CODE VALUE
         AGO   .STGFRE
.REGRC   ANOP
         LR    R2,&SYSLIST(1,1)        GET RETURN CODE VALUE
.STGFRE  ANOP
&NO24    SETC  'NO24'.'&SYSNDX'
&NO31    SETC  'NO31'.'&SYSNDX'
```

```
         CLC   BASE_24,RCØØØØ          Q. BASE 24 STORAGE PRESENT
         BNE   &NO24                   A. NONE PRESENT
         ICM   RØ,B'1111',BASE_24L
         ICM   R15,B'1111',BASE_24P
         STORAGE RELEASE,ADDR=(R1),LENGTH=(RØ),SP=(R15)
&NO24    DS    ØH
         CLC   BASE_31,RCØØØØ          Q. BASE 24 STORAGE PRESENT
         BNE   &NO31                   A. NONE PRESENT
         ICM   RØ,B'1111',BASE_31L
         ICM   R15,B'1111',BASE_31P
         STORAGE RELEASE,ADDR=(R1),LENGTH=(RØ),SP=(R15)
&NO31    DS    ØH
         SPACE 1
         L     RØ,QLENGTH              GET THE DSECT LENGTH
         SPACE 1
         STORAGE RELEASE,LENGTH=(RØ),ADDR=(R13)
         SPACE 1
         AIF   (N'&SYSLIST NE Ø).SETRC
         XR    R15,R15                 CLEAR THE RETURN CODE
         AGO   .MEND
.SETRC   ANOP
         LR    R15,R2                  SET THE RETURN CODE
.MEND    ANOP
         PR                            RETURN TO CALLER
* FOR ADDRESSABILITY PURPOSES
         LTORG
         MEND
```

$EDTEPI is a very simple macro that performs three functions. It will set the program return code. The return code can be provided in either a register or a fullword storage location. A STORAGE RELEASE will be performed for the program's dynamic storage area. We also check to see whether we have acquired any additional areas. We do this by checking the contents of BASE_24 and BASE_31 respectively. If their contents are nonzero we use the information they contain to free them as well.

$EDTSTG

To define our dynamic storage area we use the $EDTSTG macro, which is listed next.

```
         MACRO
         $EDTSTG
.*****************************************************************
```

```
.*       THE $EDTSTG MACRO IS USED TO PROVIDE A DYNAMIC WORKING       *
.*       STORAGE AREA FOR THE CURRENT CSECT.  A STANDARD REGISTER     *
.*       SAVE AREA IS PROVIDED, STORAGE AREAS FOR POINTERS TO OTHER   *
.*       ACQUIRED STORAGE AREAS, SIMPLE REGISTER SAVE AREAS AS WELL   *
.*       AS BIT PATTERNS THAT CAN BE USED FOR MASKING OPERATIONS.     *
.*       IN ADDITION, YOU CAN DEFINE YOUR OWN FIELDS AND THEY WILL    *
.*       BE INCLUDED IN THE DSECT.  THE LENGTH OF THE DSECT IS CAL-   *
.*       CULATED BY THE ASSEMBLER AND MADE AVAILABLE TO THE PROGRAM   *
.*       THROUGH THE USE OF A Q-TYPE ADDRESS CONSTANT.  IF YOU ALSO   *
.*       USE THE $EDTEPI AND $EDTPRO MACROS, THEY USE THIS QCON       *
.*       VALUE TO OBTAIN AND RELEASE THE NEEDED STORAGE.              *
.*                                                                    *
.*       EXAMPLES:                                                    *
.*                                                                    *
.*              $EDTSTG                                               *
.*      XXX     DC    F          = DEFINE ADDITIONAL STORAGE AREA     *
.*      YYY     DC    XL255                                           *
.*       .       .      .                                            *
.*       .       .      .                                            *
.*       .       .      .                                            *
.*                                                                    *
.**********************************************************************
.* DEFINE THE NEEDED GLOBAL SYMBOLS                                   *
.* THESE SYMBOLS WERE INITIALIZED IN $EDTFSU                          *
.**********************************************************************
         COPY  $EDTGLBL
.**********************************************************************
.* DEFINE LOCAL VARIABLES                                             *
.**********************************************************************
         LCLA  &NDCB
.**********************************************************************
.* DEFINE BIT PATTERN FIELDS                                          *
.**********************************************************************
##BIT0   EQU   128                    SET BIT 0 ON
##BIT1   EQU   64                     SET BIT 1 ON
##BIT2   EQU   32                     SET BIT 2 ON
##BIT3   EQU   16                     SET BIT 3 ON
##BIT4   EQU   8                      SET BIT 4 ON
##BIT5   EQU   4                      SET BIT 5 ON
##BIT6   EQU   2                      SET BIT 6 ON
##BIT7   EQU   1                      SET BIT 7 ON
.**********************************************************************
.* DEFINE RETURN CODE FILEDS                                          *
.**********************************************************************
         SPACE 1
RC0000   DC    F'0'                  USED TO SET RETURN CODES
RC0004   DC    F'4'                  USED TO SET RETURN CODES
RC0008   DC    F'8'                  USED TO SET RETURN CODES
RC000C   DC    F'12'                 USED TO SET RETURN CODES
RC0010   DC    F'16'                 USED TO SET RETURN CODES
```

```
.*********************************************************************
.* DEFINE OTHER CONSTANT FIELDS                                      *
.*********************************************************************
$$$$F1SA DC    CL4'F1SA'               USED FOR STACK OPERATIONS
$$$$4Ø96 DC    F'4Ø96'                 USED TO ADJUST BASE REGS
.*********************************************************************
.* DEFINE THE DYNAMIC AREA                                           *
.*********************************************************************
         SPACE 1
QDSECT   DC    Q(DSECT)                DEFINE A QCON
QLENGTH  CXD                           LET ASM CALCULATE THE LENGTH
DSECT    DSECT
         DS    18F                     SET ASIDE REGISTER SAVE AREA
SAR14_R1 DS    4F                      SAVE AREA FOR REGS 14, 15, Ø 1
RET_CODE DS    F                       HOLDER FOR THE CURRENT RC
UNDERTSO DS    F                       INDICATOR FIELD FOR TSO
BASE_24  DS    F                       BASE ADDRESS FOR 24 BIT STORAGE
BASE_24L DS    F                       LENGTH OF 24 BIT STORAGE
BASE_24P DS    F                       SUBPOOL NUMBER
BASE_31  DS    F                       BASE ADDRESS FOR 31 BIT STORAGE
BASE_31L DS    F                       LENGTH OF 31 BIT STORAGE
BASE_31P DS    F                       SUBPOOL NUMBER
         AIF   (&NUM_DCB_ENTRIES LT 1).MEND
&NI      SETA  1
.LOOP1   ANOP
&DYN_DCB_NAME(&NI)    DS  F
&DYN_DCBE_NAME(&NI)   DS  F
&DYN_DCB_FLAG(&NI)    DS  F
&DYN_DCB_OPEN(&NI)    DS  F
&DYN_DCB_CLOSE(&NI)   DS  F
&DYN_DCB_BFR(&NI)     DS  F
&NI      SETA  &NI+1
         AIF   (&NI LE &NUM_DCB_ENTRIES).LOOP1
.MEND    ANOP
         MEND
```

As you can see, the $EDTSTG macro provides some symbol definitions as well as storage areas. These storage areas that we have chosen to define are by no means all-inclusive. Again, you can modify the macro to add fields or remove those that you do not feel are needed. You may want to add some fields to support the new 64-bit architecture. Note that at the end of the $EDTSTG macro there is a simple loop that is used to generate fullword fields that are related to DCB definitions. You will see how those are used when we introduce some additional macros in our next article. To add fields into the dsect definition, you simply code as you normally would immediately following the $EDTSTG macro.

We will provide a simple example at the end of the article for your reference.

$EDTEND

The last macro that we will consider in this article is $EDTEND. As you might suspect from the name, this macro is placed at the end of the program source code. Its purpose is to interrogate

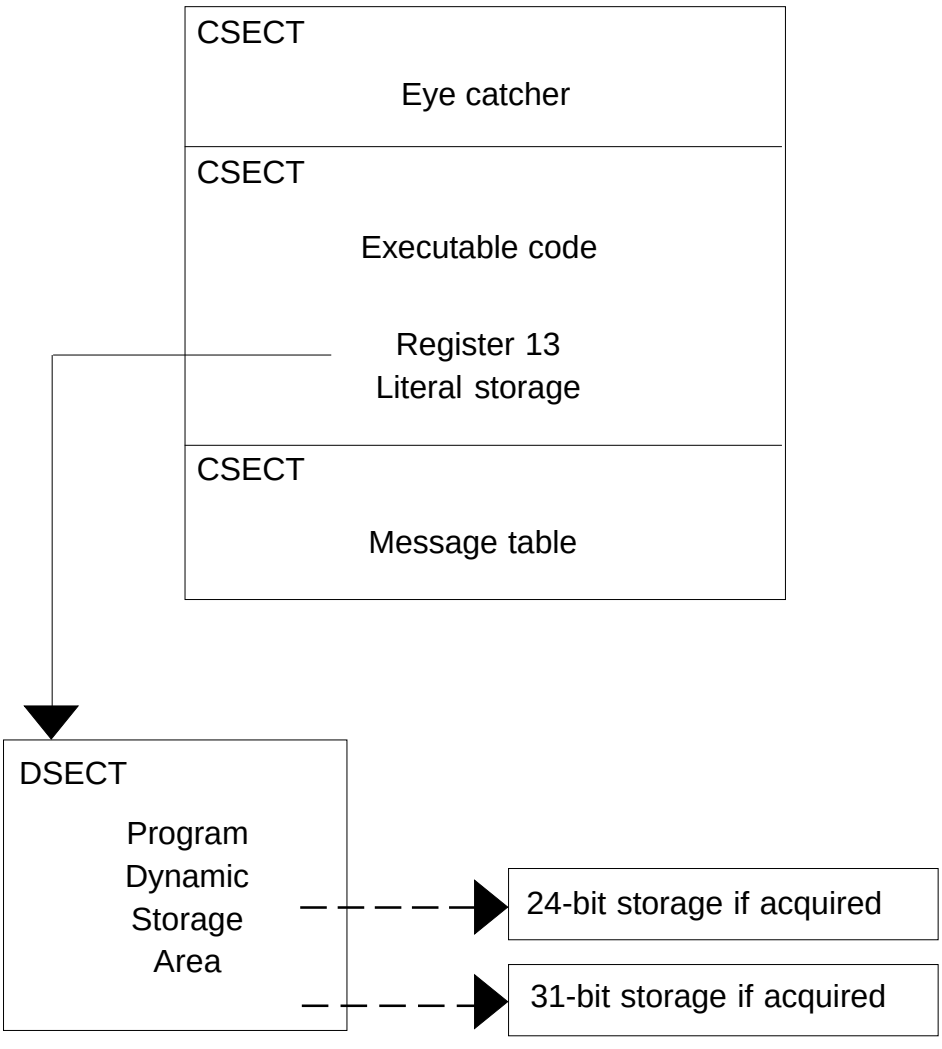

Figure 2: New module

some global symbols to determine whether additional dsects need to be included into the assembly. Lastly, it provides an end statement for the program.

```
         MACRO
         $EDTEND
.*************************************************************************
.*                                                                       *
.*       $EDTEND IS USED TO END THE PROGRAM. IT SERVERS TWO PURPOSES.    *
.*       IT CONTAINS CONDITIONAL TESTS AGAINST GLOBAL SYMBOLS TO         *
.*       WHICH IBM PROVIDED DSECTS NEED TO BE INCLUDED IN THE ASSEM-     *
.*       BLY OF THE PROGRAM.  IT ALSO INSERTS THE END STATEMENT.         *
.*                                                                       *
.*************************************************************************
         GBLC &MY_CSECT
         GBLC &PSA
         GBLC &ASCB
         GBLC &DCBD
         GBLC &DCBE
         AIF   ('&PSA' EQ 'NO').NOPSA
         IHAPSA
.NOPSA   ANOP
         AIF   ('&ASCB' EQ 'NO').NOASCB
         IHAASCB
.NOASCB  ANOP
         AIF   ('&DCBD' EQ 'NO').NODCBD
         DCBD  DSORG=(QS),DEVD=DA
.NODCBD  ANOP
         AIF   ('&DCBE' EQ 'NO').NODCBE
         IHADCBE
.NODCBE  ANOP
         END &MY_CSECT
         MEND
```

Here is a very simple program that uses the four macros that we have presented. Note that we have defined a couple of additional dynamic fields after the $EDTSTG macro to illustrate how we can add extra fields.

```
EXAMPLE  $EDTPRO
         LTR  R1,R1             Q. Anything in register 1
         BZ   ADIOS             A. No, leave the routine
         .
           Do some processing
             .
ADIOS        DS   ØH
         $EDTEPI
         $EDTSTG
COUNT1   DS   F
```

```
COUNT2   DS   F
         $EDTEND
```

In closing, we would like the reader to remember that our goal is to standardize our module structure. There is no doubt that more sophisticated macros could be developed. Our focus is not on the macros and their associated code, but on how we are using the macros to help us achieve some level of standardization.

Pictorially, our module looks like Figure 2.

Using CSI to identify VSAM datasets defined with IMBED, REPLICATE, and KEYRANGE

IBM has announced that, with z/OS 1.4, VSAM datasets defined with the attributes of IMBED, REPLICATE, and KEYRANGE will no longer be supported.

Recent versions of DFSMS have ignored the IMBED and REPLICATE attributes for the definition of new VSAM clusters; however, there is still the problem of identifying older datasets that have been defined with these attributes. Note that this may include datasets restored from back-up.

The following REXX can be used to identify (non-migrated)

VSAM clusters defined with IMBED, REPLICATE, and KEYRANGE.

```
/* Rexx */
   /* use the CSI to find VSAM datasets with     */
   /* IMBED,REPLICATE,KEYRANGE                   */
   /* probably best run in batch                 */

   work_area  = '00002000'x || copies('00'x,8192-4)
   call find_user_catalog_names
   do k = 1 to user_catalogs.0
      say 'processing 'user_catalogs.k
      call process_user_catalog
      say ' '
   end
exit
process_user_catalog :
   true = 1; false = 0
   finished = false
   modrsnrc = substr(' ',1,4)
   CSI_field = copies(' ',200)
   CSI_field = overlay('**',1)                          /* csifiltk   */
   CSI_field = overlay(user_catalogs.k,CSI_field,45)  /* csicatnm   */
   CSI_field = overlay('C',CSI_field,133)               /* csisdtyps  */
   CSI_field = overlay('Y',CSI_field,151)               /* csis1cat   */
   CSI_field = overlay('0001'x,CSI_field,153)           /* csinumen   */
   CSI_field = overlay('VSAMTYPE',CSI_field,155)      /* csifldnm 1 */
   address linkpgm 'IGGCSI00  modrsnrc  CSI_field work_area'
   csiusdln = c2d(substr(work_area,9,4))       /* used length */
   csicflg  = substr(work_area,15,1)
   if csicflg ¬= '00'x then
      signal ERROR
   csiresum = substr(CSI_field,150,1)
   i = 65   /* skip catalog info */
   do until finished
      dataset_type = substr(work_area,i+1,1)
      dataset_name = strip(substr(work_area,i+2,44),'T')
      csieflg = substr(work_area,i,1)
      if csieflg > 'BF'x then  /* error indicator */
         i = i +46 + 4
      else
         do
            if c2d(substr(work_area,i+46,2)) ¬= 8 then
               do
                  say 'ERROR value of i is 'i
                  say substr(work_area,i,60)
                  say c2x(substr(work_area,i,60))
                  signal ERROR
               end
```

```
            if dataset_type ¬= 'C' then  /* will be D or I */
               do
                  VSAM_info = substr(work_area,i+46+6,1)
                  if bitand(VSAM_info,'34'x) > 'ØØ'x then
                     do
                        say '      ' dataset_name
                        if bitand(VSAM_info,'2Ø'x) > 'ØØ'x then
                           say '       .......Imbed'
                        if bitand(VSAM_info,'1Ø'x) > 'ØØ'x then
                           say '       .......Replicate'
                        if bitand(VSAM_info,'Ø4'x) > 'ØØ'x then
                           say '       .......Keyrange'
                     end
               end
            i = i + 46 + 8
         end
      if i > csiusdln then  /* at the end of the returned info */
         do
            if csiresum ¬= 'Y' then
               finished = true
            else
               do /* get the next control blocks worth */
               address linkpgm 'IGGCSIØØ  modrsnrc  CSI_field work_area'
                  csiusdln = c2d(substr(work_area,9,4))
                  csicflg  = substr(work_area,15,1)
                  if csicflg ¬= 'ØØ'x then
                     signal ERROR
                  csiresum = substr(CSI_field,15Ø,1)
                  i = 65
               end
         end
   end
return
find_user_catalog_names : procedure expose user_catalogs. work_area
   catname = master_catalog()
   modrsnrc = substr(' ',1,4)
   CSI_field = copies(' ',2ØØ)
   CSI_field = overlay('**',CSI_field,1)          /* csifiltk   */
   CSI_field = overlay(catname,CSI_field,45)      /* csicatnm   */
   CSI_field = overlay('U',CSI_field,133)         /* csisdtyps  */
   CSI_field = overlay('Y',CSI_field,151)         /* csis1cat   */
   CSI_field = overlay('ØØØØ'x,CSI_field,153)     /* csinumen   */
   address linkpgm 'IGGCSIØØ  modrsnrc  CSI_field work_area'
   csiusdln = c2d(substr(work_area,9,4))      /* used length */
   csicflg  = substr(work_area,15,1)
   if csicflg ¬= 'ØØ'x then
      signal ERROR
   i = 65
   k = Ø
```

```
   do until i > csiusdln  /* this assumes all fit in the work area */
      k = k + 1
      user_catalogs.k = strip(substr(work_area,i+2,44),T)
      i = i + 46 + c2d(substr(work_area,i+46,2))
   end
   user_catalogs.0 = k
return
master_catalog : procedure expose work_area
   modrsnrc = substr(' ',1,4)
   CSI_field = copies(' ',200)
   CSI_field = overlay('SYS1.LINKLIB',CSI_field,1) /* csifiltk   */
   CSI_field = overlay('A',CSI_field,133)          /* csisdtyps  */
   CSI_field = overlay('Y',CSI_field,151)          /* csis1cat   */
   CSI_field = overlay('0000'x,CSI_field,153)      /* csinumen   */
   address linkpgm 'IGGCSI00  modrsnrc  CSI_field work_area'
   csicflg  = substr(work_area,15,1)
   if csicflg ¬= '00'x then
      signal ERROR
return (strip(substr(work_area,17,44),T))
```

Dave Welch (New Zealand)

As a free service to subscribers and to remove the need to rekey, the code from individual articles of *MVS Update* can be accessed on our Web site – www.xephon.com/mvs.

You will be asked to enter a word from the printed issue.

Innovation Data Processing has announced that all its enterprise software solutions are currently shipping with support for the new T9840B direct zServer FICON channel-attach tape drive introduced this week by StorageTek.

According to the company, all components of its FDR Storage Management Suite, including FDR (Fast Dump Restore), ABR, FDRINSTANT, and ABRINSTANT for non-disruptive z/OS storage protection, its FDR/UPSTREAM Storage Management Suite (including the FDRSOS and FDR/UPSTREAM/SOS non-disruptive storage management applications for Unix, Linux and Windows), DAS, SAN, LAN, and NAS distributed storage, and S/390 Unix and Linux on zServer storage, work with the new FICON T9840B drives.

Additionally, the company announced that its FATSCOPY solution for auto migrating data from older tape media like 3480/3490E cartridges to new large capacity cartridge volumes such as the T9840B while updating the customer's tape management system (CA1, IBM-RMM) to match, also fully supports the new StorageTek FICON attach tape drives.

For further information contact:
Innovation Data Processing, 275 Paterson Ave, Little Falls, NJ 07424, USA.
Tel: (973) 890 7300.
URL: http://www.innovationdp.fdr.com.

* * *

Merant has announced its Dimensions Enterprise Edition change management suite capable of supporting mainframe and distributed platforms with a single, enterprise-wide repository.

Comprising Dimensions 8 and the new Dimensions for z/OS and Merant Build, it's designed to provide a framework for automating and controlling development processes, business rules, and asset change practices across all enterprise platforms.

It captures within a single repository all metadata associated with electronic assets, processes, issues, and relationships across mainframe and distributed platforms. Authorized users can view and manage these from anywhere in the organization.

It is aimed at sites with development projects that encompass mainframe, application, and Web server components.

Dimensions 8 includes DB2 and Oracle support, integration with Visual Studio .NET 2003, improved client usability, enhanced administration tools, and integration with Merant's process asset library.

Dimensions for z/OS replaces Merant's previous mainframe client and there are new enhancements to ensure integration and consistency with all other Dimensions platforms.

Architecture improvements include the addition of a standard Dimensions listener, which replaces the Mainframe Manager and is said to simplify installation and configuration, as well as a standard API toolkit that supports the connection to Dimensions and the execution of commands.

For further information contact:
Merant, 3445 NW 211th Terrace, Hillsboro, OR 97124, USA.
Tel: (503) 645 1150.
URL: http://www.merant.com./Products/ECM/dimensions/home.asp.