179 MVS

August 2001

update

In this issue

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

North American office
Xephon/QNA
PO Box 350100,
Westminster, CO 80035-0100
USA
Telephone: (303) 410 9344
Fax: (303) 438 0290

Contributions

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, you can download a copy of our *Notes for Contributors* from www.xephon.com/contnote.html.

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/mvsupdate.html; you will need to supply a word from the printed issue.

Editor
Jaime Kaminski

Disclaimer

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

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 1992 issue, are available separately to subscribers for £29.00 ($43.00) each including postage.

Printed in England.

A REXX EDIT macro to clean up the PROG*xx*

Recently I discovered that the PROG*xx* members of some of our systems were out of sync, so, for example, some of the volumes that were specified, no longer existed. I decided I needed a foolproof method for cleaning up that member, so I wrote a simple REXX edit macro to flag the statements that were in error. This macro is intended to check for libraries that do not exist in the volumes specified, or for SMS-managed ones that do not exist in the standard catalog search order.

The macro will read all the lines in the PROG*xx* member being edited, and it will check for the APF ADD keywords. When found, it will check for a VOLUME(vvvvvv) or SMS keyword at the end of that line. It will not deal with statements spanning more than one line.

If one of those two situations is found, the macro will attempt to perform an alloc of that library with a dynamic generated DDname. That being successful, it will free the library, and it will go in search of the next statement. Otherwise it will flag that statement with a MSGLINE right after it, with the alloc return code, and a timestamp:

```
===> RC=rc (yy/mm/dd - hh:mm:ss)
```

It will also display that statement on the screen. Prior to the first statement in ‘error’, it will display the SMFID of the system. At the end of the execution, the first flagged statement, if any, will become the current line. The macro forces a return code of 1, to ensure that the cursor will be in the command line.

VRFAPF

```
/* REXX
                        ************************
                        *    VRFAPF  1.Ø.Ø     *
                        ************************
-                                                                    */
address "ISREDIT"
"macro"
"(lastline) = linenum .zlast"
x=outtrap("ON")
found=Ø
```

```
ddname="V"time("S")
do  n=1 to lastline
    "(curline) = line" n
    if  subword(space(curline),1,2)="APF ADD" then
        do
            parse value curline with "("dsn")" "("volume")"
            if  volume="" then
                do
                    if  word(curline,words(curline))="SMS" then
                         do
                            address "TSO",
                               "alloc f("ddname") shr reuse da('"dsn"')"
                            if  rc=Ø then
                               do
                                    address "TSO" "free f("ddname")"
                               end
                            else
                               do
                                    call no_good
                               end
                         end
                end
            else
                do
                    address "TSO",
                        "alloc f("ddname") shr reuse da('"dsn"')",
                            "vol("volume")"
                    if  rc=Ø then
                        do
                            address "TSO" "free f("ddname")"
                        end
                    else
                        do
                            call no_good
                        end
                end
        end
end
x=outtrap("OFF")
if  found then
    do
        "locate "lx
    end
return
/* - - - - - - - - - - */
no_good:
if  ¬found then
    do
        found=1
        lx=n
        say""
```

```
        say center(mvsvar("SYSNAME"),79)
        say""
    end
say strip(curline,"T")
"LINE_AFTER "n" = MSGLINE '===> RC="rc" ("Date("O")"-"Time()")'"
return
/* - - - - - - - - - - */
```

Joao Bentes de Jesus
Systems Programmer
Mundial-Confiana SA (Portugal)

A program to display paragraph names and remove after testing

Sometimes to understand the flow of the program, developers introduce 'DISPLAY PARAGRAPHNAME' after every paragraph. Some programs are so huge that it can take hours to add this statement, because the user needs to locate every paragraph and introduce the above statement with the paragraph name. Removing these statements is easier than introducing them. The utility program CBLDISP inserts the 'DISPLAY …' statement after every paragraph if it is not present and removes the 'DISPLAY …' statement if present.

```
/* ------------------------------------------------------- */
/*  Function:                                              */
/*  This EXEC adds or removes 'DISPLAY ParagraphName.' */
/*  after every paragraph in a COBOL source. If the    */
/*  DISPLAY statement is already present after a       */
/*  paragraph, then it removes all such statements.    */
/*  If you have to specifically remove, Use            */
/*  CBLDISP X to remove.                               */
/*  To be used in ISPF/EDIT                            */
/* ------------------------------------------------------- */

Trace O

Address ISREDIT
'MACRO (xParams) PROCESS'
'(xSaveSt) = USER_STATE'
'NUMBER = OFF'
"FIND ' PROCEDURE' 7 2Ø FIRST"
If Rc=Ø Then Do
  '(nStart) = LineNum .ZCSR'
```

```
  'CURSOR = ' nStart+1
  End
 Else Do
  'USER_STATE = (xSaveSt)'
  Say '***Error*** No PROCEDURE DIVISION Found In The Source'
  Say '            Program Aborted'
  Exit
  End

J=Ø
sProcDiv='N'
sFirstTime='Y'
Do Forever
 "FIND P'$' 8 8 NEXT"
 If Rc<>Ø Then Leave
 '(xData) = Line .ZCSR'
 '(I) = LineNum .ZCSR'
 If Rc<>Ø Then Do
   zedsmsg='***Data Error***'
   zedlmsg='***Data Error*** Data Could Not Be Read'
   Address ISPEXEC 'SETMSG MSG(ISRZØØ1)'
   Exit 1
   End
 If Substr(xData,7,1)<>'' Then Iterate
 xData=Substr(xData,8,65)
 Select
  When WordPos('EJECT',xData)>Ø Then Nop
  When WordPos('COPY',xData)>Ø Then Nop
  When Pos('—IN',xData)>Ø Then Nop
  When Pos('++IN',xData)>Ø Then Nop
  When Pos('**IN',xData)>Ø Then Nop
  Otherwise Do
   If sFirstTime='Y' Then Call DecideAddOrDel
   If xParams='' Then Do
     J=J+1
     xProcName.J=Word(xData,1) I
     End
    Else Do
     xParaName=Word(xData,1)
     K=I+1
     '(xData) = Line (K)'
     Parse Var xData 1 . 8 xPart1 xPart2
     If xPart1='DISPLAY' & "'"xParaName"'" = xPart2 Then Do
       'Delete' K
       If Rc<>Ø Then Do
         zedsmsg='***Delete Error***'
         zedlmsg='***Delete Error*** Data Could Not Be Deleted'
         Address ISPEXEC 'SETMSG MSG(ISRZØØ1)'
         Exit 1
         End
```

```
       J=J+1
       End
     End
   End
  End
 End
If xParams='' Then Do
  Do I=J By -1 Until I=1
   Parse Var xProcName.I xProcName nLine
   xDisp=Copies(' ',1Ø) 'DISPLAY' "'"xProcName"'"
   'Line_After (nLine) = (xDisp)'
   If Rc<>Ø Then Do
     zedsmsg='***Insert Error***'
     zedlmsg='***Insert Error*** Data Could Not Be Inserted'
     Address ISPEXEC 'SETMSG MSG(ISRZØØ1)'
     Exit 1
     End
   End
  xMsg1='Lines Inserted='J
  xMsg2= J 'DISPLAY Paragraph-Name statements were added to the file'
  End
 Else Do
  xMsg1='Lines Deleted='J
  xMsg2= J 'DISPLAY Paragraph-Name statements were deleted from the
file'
  End
'USER_STATE = (xSaveSt)'
zedsmsg=xMsg1
zedlmsg=xMsg2
Address ISPEXEC 'SETMSG MSG(ISRZØØ1)'
Return

DecideAddOrDel:
sFirstTime='N'
If xParams='' Then Do
  Parse Var xData xPara1 .
  '(K) = LineNum .ZCSR'
  K=K+1
  '(xDataNxt) =  Line (K)'
  Parse Var xDataNxt 1 . 12 xPart1 xPart2
  If xPart1='DISPLAY' & xPart2="'"xPara1"'" Then xParams='DEL'
  End
Return
```

Moyeen Khan
Consultant
Decision Consultants (USA)

Flashcopy status monitor for ESS

Last year we installed four IBM Enterprise Storage Server (ESS) subsystems. These boxes are officially known as 2105-F20 machines, and are also widely known under their codename 'Shark'. An ESS can be connected to the OS/390 world, as well as to the 'open environment' (Unix and/or NT). The connections can be mixed within one subsystem (part of the DASD for OS/390, another part for Unix and NT). We only connect the boxes to OS/390.

Our boxes are fully loaded with 36GB SSA drives (no expansion cabinets are used), arranged in 16 '8-packs'. This provides each subsystem with 3.6TB of usable OS/390 3390 format DASD space. These 16 8-packs are connected two-by-two in eight SSA loops. Each subsystem comes with 16GB cache. For host connection we have the maximum of 32 ESCON channels connected to each box.

For each box we have the PAV (Parallel Access Volume) feature activated, as well as the Flashcopy feature. Both features are activated for each whole subsystem. PAV is a performance feature. It basically allows multiple concurrent non-conflicting I/Os to a single DASD volume. As a result, IOSQ time (the time the UCB in OS/390 is blocked for other I/Os to the same volume because the UCB is busy) is highly reduced.

Every box is configured for eight LCUs, each LCU owning two 36GB 8-packs. This provides space for 146 real 3390-3 volumes in one LCU. The remaining 110 addresses in each LCU are used as alias addresses for the first 110 real volumes in the LCU (every real volume is assigned one alias, of upwards from X'00', as long as there are alias UCBs left). The alias UCBs are assigned downwards, starting from X'FF'. These alias addresses are used for PAV.

One short remark on the performance of the ESS: coming from 'older' type subsystems (9393 RVAs and 9392 RAMAC/3), we see an enormous performance gain, especially for caches hostile workload (DB2, IMS). The sequential throughput of the ESS also appears to be much higher, compared with older-type subsystems.

The feature which we appreciate most, however, is the Flashcopy

function, with which you can nearly instantaneously take a copy of a volume. This allows RVA snapshot-like functions in an ESS subsystem. We used the snapshot feature in our RVAs to minimize downtime for back-ups for our online subsystems. However, there are basic differences between the RVA and the ESS in the way they provide an instantaneous copy of data.

The RVA uses a series of pointer tables in the control unit to map physical back-end storage with logical 3390 volumes. Only the used tracks on a 3390 volume are backed-up in the back-end storage. When copying a volume using the snapshot function, only the pointer data is copied to map the existing volume back-end storage to the new logical 3390 volume. So the two volumes now use the same back-end storage. If data on the source or the target volume is modified, new back-end storage is allocated and used, and the pointer tables are adjusted to point to this new back-end storage. For the comparison with the Flashcopy function of the ESS, it is important to realize that no physical data movement is involved when a volume is copied with the snapshot copy function. The RVA snapshot function can be used for volumes as well as for single datasets, but only for copies within an RVA subsystem.

The ESS is not a virtual storage subsystem (IBM just announced it will not go virtual on the ESS). This means that, for every defined 3390 volume you want to copy, the physical storage must exist on the back-end drives. This implies that if you copy a volume to another volume (using ‘normal’ or Flashcopy copy technique) the physical volume space for the target volume must be available (as opposed to the RVA, where initially no extra back-end storage is occupied).

So how is this Flashcopy function implemented? Flashcopy is (for now) only available when copying volumes. When a source volume is copied to a target volume using the Flashcopy function, the target volume is (almost) immediately accessible (as soon as the Flashcopy relationship is established), and the job is finished. Now the control unit itself takes care of copying all remaining data of the source volume to the target volume. This copying runs as a background task in the ESS.

Flashcopy sessions can be started in three different ways via TSO commands (FCESTABL), via the ESS specialist (mainly for open systems Flashcopy), or via standard DFDSS functions.

The FCESTABL TSO command implies that the whole volume is copied, no matter what amount of data is really allocated.

If initiated by DFDSS, only the allocated space (at least the first 99 allocated extents) from the source volume is copied in the background process. Once the copy is initiated, the source and the target volume are said to be in a 'Flashcopy relationship'. This relationship ends when the copying of all data from the source volume to the target volume is complete.

There is a restriction in the Flashcopy function. As the copying is done over the SSA loops, the target and the source volume must be part of the same SSA loop. Because there is a one-to-one relationship between an SSA loop and an OS/390 LCU definition, the source and the target volumes must belong to the same OS/390 LCU definition. You can easily see that, if Flashcopy is to be used, some caution must be exercised in source and target volume placement in the ESS.

Another restriction is the fact that, at any one time, a volume can be part of only one 'Flashcopy relationship'. Let's take an example to illustrate the consequences of this (using DFDSS). Source volume A is copied to target volume A1. This copy is taken using the Flashcopy function. If another copy of volume A is required, this will only be taken using the Flashcopy function if the background copy function between A and A1 is complete. If the copy is not complete, the second volume copy will be done, using normal (slow) data movement.

When using DFDSS (ADRDSSU program) to copy volumes or datasets, the program queries the subsystem (via the ANTMAIN address space) to determine whether the snapshot or Flashcopy mode of copy can be used (Fast Replication Function). If it can, then the copy is done using the snapshot or the Flashcopy function, instead of physically copying the data.

DFDSS gives a message indicating that a special copy technique was used:

```
ADR806I (005)-T0MI (02), VOLUME COPIED USING A FAST REPLICATION FUNCTION
```

For the moment DFDSS supports only full-volume Flashcopy functions. However, if you look at the documentation of the native Flashcopy commands, you can see that everything is already provided for dataset-level Flashcopy.

One remark though: when copying SMS-managed volumes, DFDSS required the use of the COPYVOLID keyword, which in turn implies that the target volume goes offline (duplicate VOLSER), and needs to be reformatted to be accessed again (eg to take a back-up). This is a bit of a hassle, if you just want to use the target to take a back-up. IBM has recognized this as an issue and has come up with the DUMPCONDITIONING keyword, which allows the target volume to remain online. The keyword is introduced with APAR OW45674 and is of great value for us.

Using the TSO FCQUERY command, you can request information on the Flashcopy status of addresses in the ESS. FCQUERY is, however, not so user-friendly, so we created a small application which uses the ANTRQST to directly query the ANTMAIN address space for information on the Flashcopy status. This enabled us to check the Flashcopy status in a convenient way, for our four boxes.

A few remarks on the design of the application. As some addresses must be queried, we implemented a ‘driver’ program, which attaches the ‘query’ program in parallel for each subsystem queried. For a volume in a Flashcopy relationship with another volume, we show only the source volume in the list. The application consists of two REXX programs (IPPFCMEN and IPPFCDET), several panels and messages, and three Assembler programs (IPPFCDRV, IPPFCQRY, and IPPUCBRX). The Assembler programs are amode 31, rmode any. Volumes for which information is requested need not be online; of course, for an offline volume the VOLSER cannot be obtained.

The application gave us a better insight into the duration of the background copy, and in general into the way the Flashcopy function works. The PTF for APAR OW47323 should be applied when using this program. The menu panel is customized for our environment (addresses of the ESSs), and thus probably needs customization in your environment. This is a screenshot of the menu panel:

```
                    Flashcopy Info : Control unit selection ...
COMMAND ===>                                            SCROLL ===> CSR

Select one of the controlunits below :

    All ESS Control units (takes some time)

     ESS 1 (Axxx)      ESS 2 (Bxxx)       ESS 3 (Cxxx)      ESS 4 (Dxxx)
```

```
A000-A0FF         B000-B0FF         C000-C0FF         D000-D0FF
A100-A1FF         B100-B1FF         C100-C1FF         D100-D1FF
A200-A2FF         B200-B2FF         C200-C2FF         D200-D2FF
A300-A3FF         B300-B3FF         C300-C3FF         D300-D3FF
A400-A4FF         B400-B4FF         C400-C4FF         D400-D4FF
A500-A5FF         B500-B5FF         C500-C5FF         D500-D5FF
A600-A6FF         B600-B6FF         C600-C6FF         D600-D6FF
A700-A7FF         B700-B7FF         C700-C7FF         D700-D7FF
```

TWO REXX PROGRAMS

IPPFCMEN displays a menu from which the LCU or ESS for which FCQUERY information can be selected:

```
/* REXX                                                              */
/*                                                                   */
/* REXX PROGRAM SHOWING A SELECTION MENU FOR FCQUERY STATUS          */
/*                                                                   */
/*  ACCEPTS ONE OR MORE LCUs AS PARAMETERS WHICH ARE DIRECTLY        */
/*  PASSED TO THE IPPFCDET REXX PROGRAM                              */
/*  PARAMETERS ARE OPTIONAL                                          */
ARG CU
NUMERIC DIGITS 10
IF CU <> ''
 THEN
 DO
  CALL IPPFCDET CU
 END
 ELSE
 DO
  ALL = ''
  DO WHILE RC < 8
   ADDRESS ISPEXEC 'DISPLAY PANEL(IPPFCMEN)'
   IF RC < 5
    THEN
    DO
     CUS = ''
     IF ALL  = '/' THEN CUS = 'ALL '
      ELSE
      DO
       IF ESS1 = '/' THEN CUS = CUS'ESS1 '
        ELSE
        DO
         IF A000 = '/' THEN CUS = CUS'A000 '
         IF A100 = '/' THEN CUS = CUS'A100 '
         IF A200 = '/' THEN CUS = CUS'A200 '
         IF A300 = '/' THEN CUS = CUS'A300 '
         IF A400 = '/' THEN CUS = CUS'A400 '
         IF A500 = '/' THEN CUS = CUS'A500 '
         IF A600 = '/' THEN CUS = CUS'A600 '
```

```
          IF A700 = '/' THEN CUS = CUS'A700 '
         END
        IF ESS2 = '/' THEN CUS = CUS'ESS2 '
         ELSE
         DO
          IF B000 = '/' THEN CUS = CUS'B000 '
          IF B100 = '/' THEN CUS = CUS'B100 '
          IF B200 = '/' THEN CUS = CUS'B200 '
          IF B300 = '/' THEN CUS = CUS'B300 '
          IF B400 = '/' THEN CUS = CUS'B400 '
          IF B500 = '/' THEN CUS = CUS'B500 '
          IF B600 = '/' THEN CUS = CUS'B600 '
          IF B700 = '/' THEN CUS = CUS'B700 '
         END
        IF ESS3 = '/' THEN CUS = CUS'ESS3 '
         ELSE
         DO
          IF C000 = '/' THEN CUS = CUS'C000 '
          IF C100 = '/' THEN CUS = CUS'C100 '
          IF C200 = '/' THEN CUS = CUS'C200 '
          IF C300 = '/' THEN CUS = CUS'C300 '
          IF C400 = '/' THEN CUS = CUS'C400 '
          IF C500 = '/' THEN CUS = CUS'C500 '
          IF C600 = '/' THEN CUS = CUS'C600 '
          IF C700 = '/' THEN CUS = CUS'C700 '
         END
        IF ESS4 = '/' THEN CUS = CUS'ESS4 '
         ELSE
         DO
          IF D000 = '/' THEN CUS = CUS'D000 '
          IF D100 = '/' THEN CUS = CUS'D100 '
          IF D200 = '/' THEN CUS = CUS'D200 '
          IF D300 = '/' THEN CUS = CUS'D300 '
          IF D400 = '/' THEN CUS = CUS'D400 '
          IF D500 = '/' THEN CUS = CUS'D500 '
          IF D600 = '/' THEN CUS = CUS'D600 '
          IF D700 = '/' THEN CUS = CUS'D700 '
         END
        END
      IF CUS <> ''
       THEN
       DO
        CALL IPPFCDET CUS
       END
     END
   END
END
EXIT
```

These are the two panels used by the IPPFCMEN REXX program:

```
)ATTR
```

```
 # TYPE(OUTPUT) INTENS(LOW) CAPS(ON) JUST(LEFT)
 [ TYPE(OUTPUT) INTENS(HIGH) CAPS(OFF) JUST(LEFT) COLOR(WHITE)
 ~ TYPE(TEXT) INTENS(HIGH) CAPS(ON) JUST(LEFT) COLOR(RED)
 $ TYPE(TEXT) INTENS(HIGH) CAPS(ON) JUST(LEFT) COLOR(YELLOW)
 £ TYPE(TEXT) INTENS(HIGH) CAPS(ON) JUST(LEFT) COLOR(PINK)
 + TYPE(TEXT) INTENS(HIGH) CAPS(ON) JUST(LEFT) COLOR(WHITE)
 _ TYPE(INPUT) INTENS(HIGH) CAPS(ON) JUST(LEFT) COLOR(WHITE) PAD(' ')
 } TYPE(TEXT) INTENS(HIGH) COLOR(RED)
 { TYPE(AB)
 " TYPE(ABSL)
)ABC DESC('File')
PDC DESC('Exit') action run(return)
)ABCINIT
.ZVARS=FCQFILE
 /*  F E D C B A 9 8 7 6 5 4 3 2 1 Ø <-+-> Ø 1 2 3 4 5 6 7 8 9 A B C D E
F */
)BODY EXPAND(//)
}{ File}
"-------------------------------------
%  / /   Flashcopy Info : Control unit selection ...  / /
%COMMAND ===>_ZCMD                          +       %SCROLL ===>_AMT +
+
+Select one of the controlunits below :
+
+ _Z~All ESS Control units (takes some time)
+
+ _Z$ESS 1 (Axxx)   _Z$ESS 2 (Bxxx)    _Z$ESS 3 (Cxxx)   _Z$ESS 4 (Dxxx)
+
+  _Z£ A000-A0FF     _Z£ B000-B0FF      _Z£ C000-C0FF     _Z£ D000-D0FF
+  _Z£ A100-A1FF     _Z£ B100-B1FF      _Z£ C100-C1FF     _Z£ D100-D1FF
+  _Z£ A200-A2FF     _Z£ B200-B2FF      _Z£ C200-C2FF     _Z£ D200-D2FF
+  _Z£ A300-A3FF     _Z£ B300-B3FF      _Z£ C300-C3FF     _Z£ D300-D3FF
+  _Z£ A400-A4FF     _Z£ B400-B4FF      _Z£ C400-C4FF     _Z£ D400-D4FF
+  _Z£ A500-A5FF     _Z£ B500-B5FF      _Z£ C500-C5FF     _Z£ D500-D5FF
+  _Z£ A600-A6FF     _Z£ B600-B6FF      _Z£ C600-C6FF     _Z£ D600-D6FF
+  _Z£ A700-A7FF     _Z£ B700-B7FF      _Z£ C700-C7FF     _Z£ D700-D7FF
)INIT
.ZVARS = '(ALL ESS1 ESS2 ESS3 ESS4 +
           A000 B000 C000 D000 A100 B100 C100 D100 +
           A200 B200 C200 D200 A300 B300 C300 D300 +
           A400 B400 C400 D400 A500 B500 C500 D500 +
           A600 B600 C600 D600 A700 B700 C700 D700)'
.CURSOR = ZCMD
.HELP = IPPFCMEH
)END
```

The second panel is:

```
%-----------  Flashcopy Info Help  ---------------
%COMMAND ===>_ZCMD
```

```
+
% Primary commands : (to be entered after COMMAND ===> )
+
%  Enter any valid ISPF command .
%
% Line commands : (to be entered in front of an item)
+
%  /  :+Selects the control unit for viewing flashcopy information.
+      +Multiple / selections are allowed
+
%  C  :+Selects the control unit for online cache statistics
+      +Only the first C selection will be processed
+
+
+
)END
```

The IPPFCDET REXX is invoked from the IPPFCMEN REXX, and obtains the requested information. This is a screenshot of the Flashcopy information.

```
File  Filter                                  Active Filter : N NONE
————————————————————————————————————

                          Flashcopy info ....         LOAD TIME =Ø.775651
COMMAND ===>                                              SCROLL ===> CSR
                                                          Sorted by  SrcA

 Srcvol   SrcA  Trgvol  TrgA  Cu  St  PCt  Progress
————————————————— Ø————————5Ø————————1ØØ
 ØOFFLØ  AØØØ               ØØ  SIM ØØ
 ØOFFLØ  AØØ1               ØØ  SIM ØØ
 ØOFFLØ  AØØ2               ØØ  SIM ØØ
 ØOFFLØ  AØØ3               ØØ  SIM ØØ
 ØOFFLØ  AØØ4               ØØ  SIM ØØ
 ØOFFLØ  AØØ5               ØØ  SIM ØØ
 ØOFFLØ  AØØ6               ØØ  SIM ØØ
 ØOFFLØ  AØØ7               ØØ  SIM ØØ
 ØOFFLØ  AØØ8               ØØ  SIM ØØ
 ØOFFLØ  AØØ9               ØØ  SIM ØØ
 ØOFFLØ  AØØA               ØØ  SIM ØØ
 ØOFFLØ  AØØB               ØØ  SIM ØØ
```

IPPFCDRV

```
/* REXX */
/* REXX PROGRAM INVOKING THE IPPFCDRV PROGRAM TO OBTAIN FCQUERY DATA */
/* AND SHOW THE RESULTS.                                             */
/* INBVOKED BY THE IPPFCMEN PROGRAM                                  */
/*                                                                   */
ARG ARGCU
NUMERIC DIGITS 1Ø
```

```
ADDRESS ISPEXEC ,
 'VGET (ZSCREEN)'
CURFIL = 'NONE'
FILT = 'N'
FCQFS = ''
ADDRESS ISPEXEC 'VGET (FCQFS) SHARED'
IF RC = Ø &,
   FCQFS ¬= ''
 THEN
 DO
  FILT = 'S'
  CURFIL = FCQFS
 END
 ELSE
 DO
  ADDRESS ISPEXEC 'VGET (FCQFSV) SHARED'
  IF RC = Ø &,
     FCQFSV ¬= ''
   THEN
   DO
    FILT = 'V'
    CURFIL = FCQFSV
   END
   ELSE
   DO
    ADDRESS ISPEXEC 'VGET (FCQFTV) SHARED'
    IF RC = Ø &,
       FCQFTV ¬= ''
     THEN
     DO
      FILT = 'T'
      CURFIL = FCQFTV
     END
   END
 END
RC = Ø
SORTFLD = 'FCSADN'
SORTORD = 'A'
SELECFLD = ' '
FCQT = 'FCQT'||ZSCREEN
CALL FCQCCMDS
IF RC = Ø
 THEN
 DO
  DO WHILE RC < 8
   ADDRESS ISPEXEC 'TBDISPL ' FCQT 'PANEL(IPPFCDET)'
   IF RC < 5
    THEN
    DO
     IF ZCMD ¬= ' '
      THEN
      DO
```

```
INTERPRET 'PARSE VALUE ZCMD WITH CMD ARG ORDER'
ZCMD = ''
IF CMD = 'SRT'
 THEN
 DO
  IF ARG = 'FCSVOL' |,
     ARG = 'FCSADN' |,
     ARG = 'FCTVOL' |,
     ARG = 'FCTADN' |,
     ARG = 'FCC' |,
     ARG = 'FCS' |,
     ARG = 'FCPC' |,
     ARG = 'FCPCG'
   THEN
   DO
    SORTFLD = ARG
    SORTORD = ORDER
    IF SORTORD = 'D'
     THEN
     DO
      ADDRESS ISPEXEC 'TBSORT ' FCQT 'FIELDS('SORTFLD',C,D)'
     END
     ELSE
     DO
      ADDRESS ISPEXEC 'TBSORT ' FCQT 'FIELDS('SORTFLD')'
     END
   END
   ELSE
   DO
    SM = 'INVALID SORT '||ARG
    ADDRESS ISPEXEC 'SETMSG MSG(THINKØØ1)'
   END
 END
 ELSE
 DO
  IF CMD = 'LOC' | ,
     CMD = 'L'
   THEN
   DO
    ADDRESS ISPEXEC 'TBVCLEAR ' FCQT
    SARG = ARG
    ARG = ARG||'*'
    IF SORTFLD = 'FCSADN' |,
       SORTFLD = 'FCTADN'
     THEN
     DO
      ARG = SARG
      IF DATATYPE(ARG,'X') = Ø
       THEN
       DO
        ARG = 'ØØØØ'
       END
```

```
  END
 IF SORTFLD = 'FCSVOL' THEN FCSVOL = ARG
 IF SORTFLD = 'FCSADN' THEN FCSADN = X2D(ARG)
 IF SORTFLD = 'FCTVOL' THEN FCTVOL = ARG
 IF SORTFLD = 'FCTADN' THEN FCTADN = X2D(ARG)
 IF SORTFLD = 'FCC'    THEN FCC    = ARG
 IF SORTFLD = 'FCS'    THEN FCS    = ARG
 IF SORTFLD = 'FCPC'   THEN FCPC   = ARG
 IF SORTFLD = 'FCPCG'  THEN FCPCG  = ARG
 ADDRESS ISPEXEC 'TBTOP' FCQT
 ADDRESS ISPEXEC 'TBSCAN' FCQT 'ARGLIST('SORTFLD')' ,
                 'CONDLIST(GE)'
 IF RC ¬= Ø
  THEN
  DO
   ADDRESS ISPEXEC 'TBBOTTOM' FCQT
  END
END
ELSE
DO
 IF CMD = 'REFRESH'
  THEN
  DO
   ADDRESS ISPEXEC ,
   'TBCLOSE '  FCQT
   ADDRESS ISPEXEC ,
   'TBERASE '  FCQT
   CALL FCQCCMDS
  END
  ELSE
  DO
   IF CMD = 'FSET'
    THEN
    DO
     ADDRESS ISPEXEC "ADDPOP"
     ADDRESS ISPEXEC "DISPLAY PANEL(IPPFCFIL)"
     RETC = RC
     ADDRESS ISPEXEC "REMPOP"
     IF RETC = Ø
      THEN
      DO
       FILT = 'N'
       CURFIL = 'NONE'
       ADDRESS ISPEXEC 'VPUT (FCQFS) SHARED'
       IF FCQFS ¬= ''
        THEN
        DO
         FILT = 'S'
         CURFIL = FCQFS
        END
       ADDRESS ISPEXEC 'VPUT (FCQFSV) SHARED'
       IF FCQFSV ¬= ''
```

```
             THEN
             DO
              FILT = 'V'
              CURFIL = FCQFSV
             END
            ADDRESS ISPEXEC 'VPUT (FCQFTV) SHARED'
            IF FCQFTV ¬= ''
             THEN
             DO
              FILT = 'T'
              CURFIL = FCQFTV
             END
            RC = Ø
            CALL FCQCCMDS
           END
           ELSE
           DO
           END
         END
         ELSE
         DO
          IF CMD = 'FCLR'
           THEN
           DO
 ADDRESS ISPEXEC 'VERASE (FCQFS FCQFSV FCPFPO) SHARED'
           FILT = 'N'
           CURFIL = 'NONE'
           CALL FCQCCMDS
           RC = Ø
          END
          ELSE
          DO
           SM = 'INVALID COMMAND '||CMD
           ADDRESS ISPEXEC 'SETMSG MSG(THINKØØ1)'
          END
         END
       END
     END
   END
END
ELSE
DO
 CRP = ZTDTOP
 IF SELECFLD ¬= ' '
  THEN
  DO
   CRP = ZTDTOP
   ROWSELA = ZTDSELS
   DO FCQCI = 1 TO ROWSELA
     IF SELECFLD = 'X'
      THEN
      DO
```

```
            END
           END
          SELECFLD = ' '
          ADDRESS ISPEXEC 'TBPUT ' FCQT
          IF FCQCI < ROWSELA
           THEN
           DO
            ADDRESS ISPEXEC 'TBDISPL ' FCQT
           END
         END
        SM = ''
        ADDRESS ISPEXEC 'TBTOP ' FCQT
        ADDRESS ISPEXEC 'TBSKIP ' FCQT 'NUMBER('CRP')'
        ADDRESS ISPEXEC 'SETMSG MSG(THINKØØ1)'
       END
    END
    ELSE
    DO
     TCMD = 'BACK'
    END
  END
END
RETURN Ø
/* REXX */
FCQCCMDS:
 ADDRESS ISPEXEC "ADDPOP"
ADDRESS ISPEXEC ,
'TBCREATE '  FCQT ,
'NAMES(SELECFLD FCSVOL FCSAD FCSADN FCTVOL FCTAD FCTADN',
 'FCC FCS FCPC FCPCG)' ,
' NOWRITE REPLACE'
IF RC > 4
 THEN
 DO
  SAY 'TBCREATE FAILED ' RC
 END
 ELSE
 DO
  ADDRESS ISPEXEC 'TBVCLEAR ' FCQT
  IF RC ¬= Ø
   THEN
   DO
    SAY 'TBVCLEAR FAILED ' RC
   END
   ELSE
   DO
    IF INDEX(ARGCU,'ALL') > Ø
     THEN
     DO
      ARGCU =      'AØØØ A1ØØ A2ØØ A3ØØ A4ØØ A5ØØ A6ØØ A7ØØ '
      ARGCU = ARGCU'BØØØ B1ØØ B2ØØ B3ØØ B4ØØ B5ØØ B6ØØ B7ØØ '
      ARGCU = ARGCU'CØØØ C1ØØ C2ØØ C3ØØ C4ØØ C5ØØ C6ØØ C7ØØ '
```

```
      ARGCU = ARGCU'D000 D100 D200 D300 D400 D500 D600 D700 '
     END
    IF INDEX(ARGCU,'ESS1') > 0
     THEN
     DO
      ARGCU = 'A000 A100 A200 A300 A400 A500 A600 A700'
     END
    IF INDEX(ARGCU,'ESS2') > 0
     THEN
     DO
      ARGCU = 'B000 B100 B200 B300 B400 B500 B600 B700'
     END
    IF INDEX(ARGCU,'ESS3') > 0
     THEN
     DO
      ARGCU = 'C000 C100 C200 C300 C400 C500 C600 C700'
     END
    IF INDEX(ARGCU,'ESS4') > 0
     THEN
     DO
      ARGCU = 'D000 D100 D200 D300 D400 D500 D600 D700'
     END
    CALL LOADFCQC
   IF SORTORD = 'D'
    THEN
    DO
     ADDRESS ISPEXEC 'TBSORT ' FCQT 'FIELDS('SORTFLD',C,D)'
    END
    ELSE
    DO
     ADDRESS ISPEXEC 'TBSORT ' FCQT 'FIELDS('SORTFLD')'
    END
   ADDRESS ISPEXEC 'TBTOP ' FCQT
   END
 END
 ADDRESS ISPEXEC "REMPOP"
RETURN RC
/* REXX */
LOADFCQC:
 MESS = 'QUERYING CU 'ARGCU
 ADDRESS ISPEXEC 'CONTROL DISPLAY LOCK'
 ADDRESS ISPEXEC 'DISPLAY PANEL(IPPFCSTA)'
    PARM = ''
    DO CNT = 1 TO WORDS(ARGCU)
     CU = WORD(ARGCU,CNT)
     CUD = X2D(CU)
     CUX.CNT = D2C(CUD)
     BUFFER.CNT = RIGHT(' ',32000,' ')
     VOLTAB.CNT = RIGHT(' ',2048,' ')
     PARM = PARM||' CUX.'CNT' BUFFER.'CNT' VOLTAB.'CNT
    END
  X = TIME('R')
```

```
  INTERPRET ADDRESS LINKMVS "IPPFCDRV PARM"
 MESS = 'FORMATTING DATA FOR 'CU
 MESS = 'ELAPSED = 'TIME('E')
 ADDRESS ISPEXEC 'CONTROL DISPLAY LOCK'
 ADDRESS ISPEXEC 'DISPLAY PANEL(IPPFCSTA)'
  X = TIME('R')
 DO CNT = 1 TO WORDS(ARGCU)
  VOLTAB = VOLTAB.CNT
  BUFFER = BUFFER.CNT
  NUMVOLS = LENGTH(VOLTAB)/6
  VOLSER.Ø = NUMVOLS
  ANTDAT.Ø = NUMVOLS
  BUFFER = TRANSLATE(BUFFER,' ','#')
DO I = 1 TO NUMVOLS
   ADDRESS ISPEXEC 'TBVCLEAR ' FCQT
   FCSVOL = SUBSTR(VOLTAB,(I-1)*6+1,6)
   FCPC   = 'ØØ'
   IF FCSVOL ¬= '*FAIL*'
    THEN
    DO
     ANTDAT = TRANSLATE(WORD(BUFFER,I),'   ',',.%')
     INTERPRET 'PARSE VALUE ANTDAT WITH
         FCSAD SSID FCC ADDR CUTYPE CUSERIAL FCS REST'
     IF FCSAD = '——'
      THEN
      DO
      END
      ELSE
      DO
       FCSADN = Ø
       IF DATATYPE(FCSAD,'X')
        THEN
        DO
         FCSADN = X2D(FCSAD)
        END
       IF FCS = 'FC'
        THEN
        DO
         INTERPRET 'PARSE VALUE REST WITH
              FCPC TARG SSID FCC2 FCTAD CUTYPE CUSERIAL FCS2 REST2'
         FCTAD = LEFT(FCSAD,2)||FCTAD
         FCTADN = Ø
         IF DATATYPE(FCTAD,'X')
          THEN
          DO
           FCTADN = X2D(FCTAD)
           SAD = D2C(FCTADN)
           TVOLSR = RIGHT(' ',1Ø,' ')
           ADDRESS LINKMVS "IPPUCBRX SAD TVOLSR"
           IF RIGHT(TVOLSR,4) = FCTAD
            THEN
            DO
             FCTVOL = LEFT(TVOLSR,6)
```

```
            END
            ELSE
            DO
             FCTVOL = '*OFFL*'
            END
          END
         IF DATATYPE(FCPC) = 'NUM'
          THEN
          DO
           FCAST = LEFT('*',40,'*')
           RD = TRUNC((FCPC + 2)/3,0);
           FCPCG = LEFT(FCAST,RD)||'>'
           IF FCPC < 10
            THEN
            DO
             FCPC = '0'||FCPC
            END
          END
        END
       CALL SELTBADD
      END
     END
    END
  END
 SM = 'LOAD TIME ='TIME('E')
 ADDRESS ISPEXEC 'SETMSG MSG(THINK001)'
RETURN
/* REXX */
SELTBADD:
 LF = LENGTH(CURFIL)
 SELECT;
  WHEN FILT = 'S' THEN
   IF LEFT(FCS,LF) = CURFIL THEN ADDRESS ISPEXEC 'TBADD ' FCQT
  WHEN FILT = 'V' THEN
   IF LEFT(FCSVOL,LF) = CURFIL THEN ADDRESS ISPEXEC 'TBADD ' FCQT
  WHEN FILT = 'T' THEN
   IF LEFT(FCTVOL,LF) = CURFIL THEN ADDRESS ISPEXEC 'TBADD ' FCQT
  WHEN FILT = 'N' THEN
   ADDRESS ISPEXEC 'TBADD ' FCQT ' MULT(1024)'
 END
RETURN
```

IPPFCDET

The panels IPPFCDET, IPPFCDEH, and IPPFCFIL are shown below. IPPFCDET displays the FCQUERY output:

```
)ATTR
 £ TYPE(OUTPUT) INTENS(HIGH) CAPS(OFF) JUST(LEFT) COLOR(PINK)
 # TYPE(OUTPUT) INTENS(LOW) CAPS(ON) JUST(LEFT)
 [ TYPE(OUTPUT) INTENS(HIGH) CAPS(OFF) JUST(LEFT) COLOR(WHITE)
```

```
 ~ TYPE(OUTPUT) INTENS(HIGH) CAPS(ON) JUST(LEFT) COLOR(RED)
 $ TYPE(TEXT) INTENS(HIGH) CAPS(ON) JUST(LEFT) COLOR(YELLOW)
 + TYPE(TEXT) INTENS(HIGH) CAPS(ON) JUST(LEFT) COLOR(WHITE)
 _ TYPE(INPUT) INTENS(HIGH) CAPS(ON) JUST(LEFT) COLOR(WHITE) PAD(' ')
 } TYPE(TEXT) INTENS(HIGH) COLOR(RED)
 { TYPE(AB)
 " TYPE(ABSL)
)ABC DESC('File')
PDC DESC('Exit') action run(return)
)ABCINIT
.ZVARS=FCQFILE
)ABC DESC('Filter')
PDC DESC('Set ...')    action run(FSET)
PDC DESC('Clear')      action run(FCLR)
)ABCINIT
.ZVARS=FCQFILT
)ABCPROC
VER (&FCQFILT,RANGE,1,2)
 /*  F E D C B A 9 8 7 6 5 4 3 2 1 Ø <-+-> Ø 1 2 3 4 5 6 7 8 9 A B C D E F */
)BODY EXPAND(//)
}{ File{ Filter}                              Active Filter :[Z[CURFIL
"-------------------------------------------------------------------------------
%  / /   Flashcopy info ....   / /
%COMMAND ===>_ZCMD                                +           %SCROLL ===>_AMT
+
+                                                            +Sorted by
£SORTFLDP+
+[TSVOL +[TSAD+[TTVOL +[TTAD+[TC+[TS+[TPC+[TPCG
+------------------ Ø-------5Ø-------1ØØ
)MODEL
 #Z      #Z    #Z      #Z    #Z  #Z  #Z   #Z
)INIT
IF (&SORTFLD = 'FCSADN')
 &SORTFLDP = 'SrcA'
IF (&SORTFLD = 'FCSVOL')
 &SORTFLDP = 'Srcvol'
IF (&SORTFLD = 'FCTVOL')
 &SORTFLDP = 'Trgvol'
IF (&SORTFLD = 'FCTAD')
 &SORTFLDP = 'TrgA'
IF (&SORTFLD = 'FCC')
 &SORTFLDP = 'Cu'
IF (&SORTFLD = 'FCS')
 &SORTFLDP = 'St'
IF (&SORTFLD = 'FCPC')
 &SORTFLDP = 'PCt'
IF (&SORTFLD = 'FCPCG')
 &SORTFLDP = 'Progress'
&TSVOL = 'Srcvol'
&TSAD  = 'SrcA'
&TTVOL = 'Trgvol'
&TTAD  = 'TrgA'
```

```
&TC    = 'Cu'
&TS    = 'St'
&TPC   = 'PCt'
&TPCG  = 'Progress'
.ZVARS = '(FILT FCSVOL FCSAD FCTVOL FCTAD FCC FCS FCPC FCPCG)'
&SELECFLD = &Z
.CURSOR = ZCMD
.HELP = IPPFCDEH
&AMT = CSR
)PROC
IF (.CURSOR = 'TSVOL')
 &ZCMD = 'SRT FCSVOL'
IF (.CURSOR = 'TSAD')
 &ZCMD = 'SRT FCSADN'
IF (.CURSOR = 'TTVOL')
 &ZCMD = 'SRT FCTVOL'
IF (.CURSOR = 'TTAD')
 &ZCMD = 'SRT FCTADN'
IF (.CURSOR = 'TC')
 &ZCMD = 'SRT FCC'
IF (.CURSOR = 'TS')
 &ZCMD = 'SRT FCS'
IF (.CURSOR = 'TPC')
 &ZCMD = 'SRT FCPC D'
IF (.CURSOR = 'TPCG')
 &ZCMD = 'SRT FCPC D'
)END
```

IPPFCDEH

IPPFCDEH is the help panel for IPPFCDET:

```
%———————  Flashcopy Info Help  ————————————————-
%COMMAND ===>_ZCMD
+
% Primary commands : (to be entered after COMMAND ===> )
+
%  L value    :+Locates the first "value" of the row on which the table
%              +is sorted.
%  REFRESH    :+Refreshes the data requested from the control unit ...
%
% Sorting the display :
%
%  SORT field :+Tabbing to the column header field + ENTER does the sort
%
+
)END
```

IPPFCFIL

IPPFCFIL displays the filter panel for displaying only selected rows:

```
)BODY WINDOW(60,8) EXPAND([[)
%[-[  Flashcopy - Enter filtering characteristics  [-[
%COMMAND ===>_ZCMD
+
+Status   %===>_FCQFS +                        +[ [
+Srcvol   %===>_FCQFSV+                        +[ [
+Trgvol   %===>_FCQFTV+                        +[ [
+
%-[-[-
)INIT
 .CURSOR = FCQFS
)PROC
 IF (&FCQFS ¬= &Z)
  &FCQFSV = &Z
  &FCQFTV = &Z
 ELSE
  IF (&FCQFSV ¬= &Z)
   &FCQFTV = &Z
)END
```

While processing the requests, a progress window is shown, with information on the elapsed time for the query.

```
)ATTR DEFAULT(%?_)
 £ TYPE(OUTPUT) INTENS(HIGH) CAPS(ON) JUST(LEFT) COLOR(PINK)
 # TYPE(OUTPUT) INTENS(LOW) CAPS(ON) JUST(LEFT)
 $ TYPE(OUTPUT) INTENS(HIGH) CAPS(OFF) JUST(LEFT) COLOR(WHITE)
 ~ TYPE(OUTPUT) INTENS(HIGH) CAPS(ON) JUST(LEFT) COLOR(RED)
)BODY WINDOW(60,5) EXPAND([[)
%+[-[  Flashcopy - Work in progress ...  [-[+
?][ []
?][ [~MESS                          ?[ []
?][ []
%+[-[+
)END
```

For error reporting one ISPF message is used:

```
THINK001  '&SM'
'&LM'

THINK002  '&SM'  .ALARM = YES
'&LM'
```

Now for the Assembler programs. Each program is well documented so the flow should be easy to follow. The first program is IPPFCDRV, which attaches one IPPFCQRY program for each LCU requested.

IPPFCDRV

```
IPPFCDRV TITLE 'IPPFCDRV - STARTS SUBTASKS FOR FCQUERY REQUESTS'
         SPACE 2
IPPFCDRV AMODE 31
IPPFCDRV RMODE ANY
         SPACE 2
* ------------------------------------------------------------------------
* THIS PROGRAM ATTACHES ONE IPPFCQRY INSTANCE PER LCU QUERIED'
*
* PARAMETERS : THIS PROGRAM IS CALLED FROM REXX USING THE
*  "LINKMVS" HOST ENVIRONMENT - PARAMS ARE SET UP ACCORDINGLY
*  REMARK : PARAMETERS ARE BOTH INPUT AND OUTPUT
*  PARAMETERS ARE PASSED IN TRIPLETS (ONE TRIPLET PER LCU QUERIED)
*  SEE IPPFCQRY PROGRAM FOR PARAMETER LAYOUT ...
* WE ALLOW A MAXIMUM OF 32 PARALLEL LCUS QUERIED
* ------------------------------------------------------------------------
         SPACE 2
* ------------------------------------------------------------------------
*      AS AN ATTACH MACRO EXPANDS ITS PARAMETER LISTS INLINE, WE
*      USE THIS MACRO TO EASILY COPY ATTACH CODE
* ------------------------------------------------------------------------
         SPACE 2
         MACRO
         IPPFCATT
         L     RDEVA,PDEVA          LOAD 1ST PARAM (START DEVICE)
         L     RBUFA,PBUFA          LOAD 2ND PARAM (FCQUERY DATA)
         L     RVOLA,PVOLA          LOAD 3RD PARAM (VOLUME TABLE)
         ATTACH EP=IPPFCQRY,                                          X
               ECB=(RECB),PARAM=((RDEVA),(RBUFA),(RVOLA)),            X
               VL=1
         LTR   R15,R15              TEST IF ALL WENT WELL
         BNZ   ERRATT               NO, GET OUT
         ST    R1,Ø(RTCB)           KEEP TCB ADDRESS FOR DETACH
         TM    PVOLA,X'8Ø'          CHECK IF LAST PARAMETER PROCESSED
         BO    WAIT                 GO WAIT FOR SPAWNED TASKS  ...
         BCTR  RW,RØ                ONE MORE
         LTR   RW,RW                MAXIMUM TASKS REACHED
         BZ    MAXATT               TELL THE PEOPLE AND WAIT ...
         LA    RPARAM,12(RPARAM)    POINT TO NEXT TRIPLET
         LA    RTCB,4(RTCB)         NEXT TCB SLOT
         LA    RECB,4(RECB)         NEXT ECB SLOT
         MEND
         SPACE 2
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
R11      EQU   11
R12      EQU   12
R13      EQU   13
R14      EQU   14
R15      EQU   15
         SPACE 2
RPARAM   EQU   3
RECB     EQU   4
RTCB     EQU   5
RDEVA    EQU   6
RBUFA    EQU   7
RVOLA    EQU   8
RET      EQU   9
RW       EQU   1Ø
         EJECT
IPPFCDRV CSECT
         PRINT GEN
         SPACE 2
* ------------------------------------------------------------------
*        STANDARD INIT ROUTINE
* ------------------------------------------------------------------
         SPACE 2
INITIAL  DS    ØH
         SAVE  (14,12),,IPPFCDRV.&SYSDATE.&SYSTIME
         LR    R12,R15               LOAD START ADDRESS
         LA    R11,1(R12)
         LA    R11,X'FFF'(R11)
         USING IPPFCDRV,R12,R11      ESTABLISH ADDRESSABILITY
         LA    R5,WORKAREA           ADDRESS WORKAREA
         ST    R5,8(,R13)            STORE FORW POINTER IN CALLING PGM
         ST    R13,4(,R5)            STORE BACKW POINTER IN CALLED(THIS
         LR    R13,R5
         USING WORKAREA,R13
         B     START
         EJECT
* ------------------------------------------------------------------
*        START PROCESSING
* ------------------------------------------------------------------
         SPACE 2
START    DS    ØH
         LR    RPARAM,R1           GET PARAMETER ADDRESS
         USING PARAMDS,RPARAM
         MVI   TCBAREA,X'ØØ'       INIT THE TCBAREA
         MVC   TCBAREA+1(127),TCBAREA
         MVI   ECBAREA,X'ØØ'       INIT THE ECBAREA
         MVC   ECBAREA+1(127),ECBAREA
         LA    RTCB,TCBAREA
         LA    RECB,ECBAREA
         LA    RW,32               MAX NUMBER OF PARALLEL TASKS ...
* ------------------------------------------------------------------
```

```
*        AS AN ATTACH MACRO EXPANDS ITS PARAMETER LISTS INLINE, WE
*        NEEDED TO INCLUDE THE ATTACH CODE 32 TIMES
* ---------------------------------------------------------------------
         IPPFCATT                  ATTACH IPPFCQRY
         IPPFCATT
         IPPFCATT
         IPPFCATT
         IPPFCATT
         IPPFCATT
         IPPFCATT
         IPPFCATT
         IPPFCATT
         IPPFCATT
         IPPFCATT
         IPPFCATT
         IPPFCATT
         IPPFCATT
         IPPFCATT
         IPPFCATT
         IPPFCATT
         IPPFCATT
         IPPFCATT
         IPPFCATT
         IPPFCATT
         IPPFCATT
         IPPFCATT
         IPPFCATT
         IPPFCATT
         IPPFCATT
         IPPFCATT
         IPPFCATT
         IPPFCATT
         IPPFCATT
         IPPFCATT
         IPPFCATT
MAXATT   WTO   'IPPFCDRV - MORE THAN 32 LCUS',ROUTCDE=(11),DESC=(7)
WAIT     LA    RECB,ECBAREA         ADDRESS ECB AREA
         LA    RTCB,TCBAREA         ADDRESS TCB AREA
         LA    RW,32                MAX # OF ATTACHED TASKS
WAITNEXT CLC   =F'Ø',Ø(RTCB)        TILL ALL TCBS PROCESSED
         BE    ALLENDED             LETS STOP
         LTR   RW,RW                OR MAX PROCESSED
         BZ    ALLENDED             LETS STOP
         WAIT  ECB=(RECB)           WAIT ON ECB POSDTED BY IPPFCQRY
         DETACH (RTCB)              DETACH IPPFCQRY SUBTASK
         LA    RECB,4(RECB)         POINT TO NEXT ECB
         LA    RTCB,4(RTCB)         POINT TO NEXT TCB
         BCTR  RW,RØ                ONE MORE PROCESSED
         B     WAITNEXT             LETS CHECK THE NEXT ONE
ALLENDED B     RETURN
         EJECT
* ---------------------------------------------------------------------
```



```
*        ERROR ATTACHING READ SUBTASK
* ------------------------------------------------------------------
         SPACE 2
ERRATT   DS    ØH
         ABEND Ø,DUMP                FIGURE OUT WHAT HAPPENED
         WTO   'IPPFCDRV - ERROR ATTACHING IPPFCDRV SUBTASK',          X
               ROUTCDE=(11),DESC=(7)
         B     RETURN
         EJECT
* ------------------------------------------------------------------
*        STANDARD RETURN SEQUENCE
* ------------------------------------------------------------------
         SPACE 2
RETURN   DS    ØH
         L     R13,SAVEAREA+4        GET BACK ...
         RETURN (14,12),RC=(15)
         EJECT
         DROP  R12,R11
* ------------------------------------------------------------------
*        SAVE AND WORKAREAS
* ------------------------------------------------------------------
         SPACE 2
WORKAREA DS    ØF
SAVEAREA DS    18F
         DC    CL4'TCBA'
TCBAREA  DS    32F               TCB AREA : 32 FULLWORDS
         DS    ØF
         DC    CL4'ECBA'
ECBAREA  DS    32F               ECB AREA : 32 FULLWORDS
         LTORG
         EJECT
PARAMDS  DSECT
PDEVA    DS    F
PBUFA    DS    F
PVOLA    DS    F
         END   IPPFCDRV
```

IPPFCQRY

The IPPFCQRY program uses the ANTRQST macro to query the ANTMAIN address space for Flashcopy information:

```
IPPFCQRY TITLE 'IPPFCQRY - QUERY FLASHCOPY STATUS FOR AN LCU'
         SPACE 2
IPPFCQRY AMODE 31
IPPFCQRY RMODE ANY
         SPACE 2
* ------------------------------------------------------------------
* THIS PROGRAM COLLECTS FLASHCOPY INFO FOR A LCU:
* PARAMETERS : THIS PROGRAM IS CALLED FROM IPPFCDRV USING THREE
*  PARAMETERS (THESE PARAMETERS WERE PASSED FROM A REXX PROGRAM TO
*  THE IPPFCDRV PROGRAM)
```

```
*  THE PROGRAM ISSUES A REQUEST TO ANTMAIN FOR FLASHCOPY INFORMATION
* PARAM1 : LENGTH  DS H   (INPUT VALUE IS 4)
*          CUA     DS H   (INPUT CUA IN BINARY)
*                          (OUTPUT CAN BE ERROR CODE)
* PARAM2 : LENGTH  DS H   (INPUT VALUE IS MAX LENGTH OF BUFFER)
*                          (OUTPUT VALUE IS LENGTH RETURNED)
*          BUFFER  DS C   (BUFFER CONTAINING RESULT FROM ANTRSQT CALL)
* PARAM3 : LENGTH  DS H   (INPUT VALUE IS MAX LENGTH OF VOLSERS)
*                          (OUTPUT VALUE IS LENGTH RETURNED)
*          VOLSERS DS C   (BUFFER CONTAINING VOLSERS FOR DASD QUERIED)
* ------------------------------------------------------------------------
         SPACE 2
         EJECT
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
R11      EQU   11
R12      EQU   12
R13      EQU   13
R14      EQU   14
R15      EQU   15
         SPACE 2
RDEVDS   EQU   2
RBUFDS   EQU   2
RVOLDS   EQU   2
         SPACE 2
RBUFLEN  EQU   3
RBUFFER  EQU   4
RVOLLEN  EQU   5
RVOLTAB  EQU   6
RUCB     EQU   7
         SPACE 2
RPARM    EQU   8
RET      EQU   9
RW       EQU   1Ø
RW2      EQU   11
         SPACE 2
IPPFCQRY CSECT
         PRINT GEN
         SPACE 2
*        STANDARD INIT ROUTINE
         SPACE 2
INITIAL  DS    ØH
         SAVE  (14,12),,IPPFCQRY.&SYSDATE.&SYSTIME
         LR    R12,R15               LOAD START ADDRESS
```

```
         LR    RPARM,R1
         USING IPPFCQRY,R12          ESTABLISH ADDRESSABILITY
         LA    R5,WORKAREA           ADDRESS WORKAREA
         ST    R5,8(,R13)            STORE FORW POINTER IN CALLING PGM
         ST    R13,4(,R5)            STORE BACKW POINTER IN CALLED(THIS
         LR    R13,R5
         USING WORKAREA,R13
         B     START
         EJECT
*        START PROCESSING
         SPACE 2
START    DS    ØH
         USING PARMDS,RPARM          ADDRESS THE PARAMETERS PASSED
         L     RDEVDS,ADEVDS         LOAD ADDRESS FOR STARTING UCB
         USING DEVDS,RDEVDS          ADDRESS STARTING UCB
         MVC   DEVN,STRTDEV
         DROP  RDEVDS
         L     RBUFDS,ABUFDS         ADDRESS BUFFER FOR FCQUERY DATA
         USING BUFDS,RBUFDS
         LH    RBUFLEN,BUFLEN        LOAD INPUT LENGTH
         SH    RBUFLEN,=H'128'       KEEP CUSHION SIZE TO PREVENT OVFL
         STH   RBUFLEN,BUFLENWS
         LA    RBUFFER,BUFFER
         DROP  RBUFDS
         L     RVOLDS,AVOLDS         ADDRESS THE VOLSER TABLE
         USING VOLDS,RVOLDS
         LH    RVOLLEN,VOLLEN        LOAD INPUT LENGTH
         SH    RVOLLEN,=H'6'         SUBTRACT CUSHION SIZE
         STH   RVOLLEN,VOLLENWS
         LA    RVOLTAB,VOLTAB
         DROP  RVOLDS
         XC    UCBWAREA,UCBWAREA     CLEAR UCB WORK AREA WITH BIN ZEROS
         XC    UCBAREA,UCBAREA       CLEAR UCB AREA WITH BIN ZEROS
         LA    RW,256                MAX 256 FCREQUEST PER LCU
         MVC   XDEVN,DEVN            SAVE INPUT DEVICE NUMBER
         XR    RVOLLEN,RVOLLEN
         XR    RBUFLEN,RBUFLEN
* OBTAIN UCB TO OBTAIN VOLSER FOR STARTING UCB ADDRESS
         UCBSCAN COPY,WORKAREA=UCBWAREA,UCBAREA=UCBAREA,RANGE=ALL,     X
               DEVCLASS=DASD,DEVN=DEVN,DEVNCHAR=DEVNCHAR,DYNAMIC=YES,  X
               CMXTAREA=UCBCMXT,DCEAREA=UCBDCE,DCELEN=UCBDCEL
         MVI   RCR15,ZERO
* FOLLOWING CODE CHECKS TO SEE IF THE OBTAINED UCB EQUALS THE REQUESTED
*  UCB, IF NOT SOMETHING WEIRD HAPPENED AND WE STOP ...
         MVC   DEVNWORK(2),XDEVN     TRANSFORM DEVICE NUMBER ...
         MVI   DEVNWORK+2,X'ØC'       TO CHARACTER ...
         UNPK  DEVNUNPK(5),DEVNWORK    FORMAT ...
         TR    DEVNUNPK,TRTAB
         CLC   DEVNCHAR,DEVNUNPK     COMPARE WITH DEVICE FROM UCBSCAN
         BE    LOOPSTRT
         MVI   RCR15,X'Ø4'           THIS IS DEFINITELY AN ERROR ...
         WTO   'IPPFCQRY - DEVN WRONG 1',ROUTCDE=(11),DESC=(7)
         B     LOOPEND               SKIP THE LOOP ...
```

```
LOOPSTRT DS    ØH
         CLI   RCR15,ZERO            UNTIL ALL UCB'S PROCESSED
         BNE   LOOPEND               END REACHED OR ERROR OCCURRED ...
         LTR   RW,RW                 OR MAX DEVICES (256) REACHED
         BZ    LOOPEND               OUT ...
         LA    RUCB,UCBAREA          LOAD ADDRESS UCB RETURNED BY SCAN
         USING UCBOB,RUCB            EST ADDRESSABILITY
         MVC   XQRYSIZE,=AL2(512)    SET INPUT PARAMS FOR ANTRQST MACRO
         MVC   XASYNCH,=CL3'YES'     ASYNCH REQUEST
         XC    XECB,XECB             CLEAR THE WAIT ECB
         ANTRQST ILK=ESSRVCS,                                          X
               REQUEST=FCQUERY,                                        X
               DEVN=XDEVN,                                             X
               QRYSIZE=XQRYSIZE,                                       X
               QRYINFO=XQRYINFO,                                       X
               RETINFO=XRETINFO,                                       X
               ASYNCH=XASYNCH,                                         X
               ECB=XECB,                                               X
               RETCODE=RTNCD,                                          X
               RSNCODE=RSNCD,                                          X
               MF=(E,P_LIST)
* ANTRQST CAN RETURN ERROR INFO IN DIFFERENT WAYS ... VIA THE RTNCD AND
*  THE RSNCD FIELDS, OR VIA THE RTC AND RSC FIELDS IN THE XRETINFO AREA
*
* CHECK IF THE REQUEST WAS SUCCESSFULLY STARTED ...
         CLC   RTNCD,=F'Ø'           WAS REQUEST STARTED SUCCESSFULLY
         BE    WAITECB               WAIT FOR THE RESULTS
         MVC   RTCD,RTNCD            SAVE ERROR AND REASON CODE
         MVC   RSCD,RSNCD             IN COMMON ERROR FIELDS ...
         WTO   'IPPFCQRY - RTNCD NOT Ø',ROUTCDE=(11),DESC=(7)
         B     CHECKRTC              CHECK THE ERROR CODES ...
* AS THE ANTRQST REQUEST IS ASYNCHRONOUS WE NEED TO WAIT ON THE ECB ...
WAITECB  WAIT  ECB=XECB              WAIT FOR ANTRQST TO END ...
         NC    XECB,=XL4'3FFFFFFF'   CHECK IF ALL WENT WELL
         CLC   XECB,=F'Ø'            ALL ZEROES
         BE    WAITOK                MEANS WAIT OK ...
         MVC   RTCD,XECB             SAVE ECB FIELD AS ERROR CODE
         MVC   RSCD,=F'Ø'             NO REASON CODE
         WTO   'IPPFCQRY - ECB NOT OK',ROUTCDE=(11),DESC=(7)
         B     CHECKRTC
WAITOK   MVC   RTCD,RTC            SAVE ERROR AND REASON PROVIDED IN
         MVC   RSCD,RSC             XRETINFO FIELD ...
* ANTRQST GIVES A NON-ZERO RETURN CODE INDICATING THAT THE DATA AREA
*  PROVIDED IN THE PROGRAM WAS LARGE ENOUGH ...
         CLC   RTC,=A(RQST_FCQUERY_QRYINFO_LARGE_ENOUGH)
         BE    CHECKRTC
         CLC   RTC,=F'Ø'           ANY OTHER RETCODE IS BAD ....
         BE    CHECKRTC
         MVI   RCR15,X'Ø8'         INDICATE ERROR
         B     LOOPEND
CHECKRTC DS    ØH
         CLC   RTCD,=A(RQST_PC_NUMBER_ZERO) MEANS ANTMAIN NOT ACTIVE
```

```
         BNE   CHECKRT2
         MVI   RCR15,X'2Ø'         SET ERROR ... GET OUT
         B     LOOPEND
CHECKRT2 DS    ØH
         L     RW2,RSC             ALL WENT WELL, RSC = SIZE RETURNED
         CH    RW2,=H'128'         LESS THAN CUSHION SIZE
         BNH   MOVEDATA
         LH    RW2,=H'128'         LOAD CUSHION SIZE
         MVI   RCR15,X'ØC'         INDICATE TRUNCATION
MOVEDATA DS    ØH
         BCTR  RW2,RØ              MINUS ONE FOR EXECUTE ...
         EX    RW2,MOVEBUF         MOVE THE ANTRQST PROVIDED INFO
         LA    RW2,1(RW2)          GET THE REAL LENGTH MOVED
         AR    RBUFFER,RW2         POINT FURTHER IN THE BUFFER
         AR    RBUFLEN,RW2         TOTAL LENGTH OF BUFFER
         MVC   Ø(6,RVOLTAB),UCBVOLI FILL THE VOLSER ...
         TM    UCBSTAT,UCBONLI     IS DEVICE ONLINE
         BO    MOVEVOL
         MVC   Ø(6,RVOLTAB),=C'ØOFFLØ' INDICATE VOLUME IS OFFLINE
MOVEVOL  DS    ØH
         LA    RVOLTAB,6(RVOLTAB)  POINT FURTHER IN THE BUFFER
         LA    RVOLLEN,6(RVOLLEN)  TOTAL LENGTH OF BUFFER
GETNEXT  DS    ØH
         MVI   Ø(RBUFFER),C'#'     MARK END OF DATA
         LA    RBUFLEN,1(RBUFLEN)
         LA    RBUFFER,1(RBUFFER)
         BCTR  RW,RØ               ONE MORE PROCESSED
         XR    RW2,RW2             GET NEXT DEVICE NUMBER
         ICM   RW2,B'ØØ11',XDEVN
         LA    RW2,1(RW2)
         STCM  RW2,B'ØØ11',XDEVN
         UCBSCAN COPY,WORKAREA=UCBWAREA,UCBAREA=UCBAREA,RANGE=ALL,     X
               DEVCLASS=DASD,DEVN=DEVN,DEVNCHAR=DEVNCHAR,DYNAMIC=YES,  X
               CMXTAREA=UCBCMXT,DCEAREA=UCBDCE,DCELEN=UCBDCEL
         CH    R15,=H'Ø'           UCBSCAN WAS FINE ??
         BE    CHECKDEV
         STC   R15,RCR15           STORE RC TO END THE LOOP
         B     LOOPEND             STOP IT ...
CHECKDEV DS    ØH
         MVC   DEVNWORK(2),XDEVN
         MVI   DEVNWORK+2,X'ØC'
         UNPK  DEVNUNPK(5),DEVNWORK
         TR    DEVNUNPK,TRTAB
         CLC   DEVNCHAR,DEVNUNPK   IF WE NOW HAVE A MISMATCH ...
         BE    DEVNOK2              WE WILL STOP PROCESSING
         XR    RW,RW GET OUT
         B     LOOPEND
DEVNOK2  DS    ØH
         CH    RBUFLEN,BUFLENWS    CHECK FOR OUTPUT BUFFER OVERFLOW
         BNL   BUFFFULL
         CH    RVOLLEN,VOLLENWS
         BL    LOOPSTRT
BUFFFULL MVI   RCR15,X'1Ø'
```

```
LOOPEND  DS    ØH
         L     RDEVDS,ADEVDS       WE ARE DONE, PREPARE OUTPUT
         USING DEVDS,RDEVDS
         MVC   STRTDEV,=H'Ø'       STORE RETURN CODE
         MVC   STRTDEV+1(1),RCR15
         DROP  RDEVDS
         L     RBUFDS,ABUFDS
         USING BUFDS,RBUFDS
         STH   RBUFLEN,BUFLEN      STORE BUFFER LENGTH
         DROP  RBUFDS
         L     RVOLDS,AVOLDS
         USING VOLDS,RVOLDS
         STH   RVOLLEN,VOLLEN      STORE VOLTAB LENGTH
         DROP  RVOLDS
* ------------------------------------------------------------------
*        STANDARD RETURN SEQUENCE
* ------------------------------------------------------------------
         SPACE 2
RETURN   DS    ØH
         XR    R15,R15
         L     R13,SAVEAREA+4        LOAD ADDRESS WHERE PREV REG STORED
         RETURN (14,12),RC=(15)
         EJECT
MOVEBUF  MVC   Ø(*-*,RBUFFER),XQRYINFO
         DROP  R12
* ------------------------------------------------------------------
*        SAVE AND WORKAREAS
* ------------------------------------------------------------------
         SPACE 2
WORKAREA DS    ØF
SAVEAREA DS    18F
BUFLENWS DS    H
VOLLENWS DS    H
UCBWAREA DS    CL1ØØ
         SPACE 2
UCBAREA  DS    CL48                  THE UCB AREA
         SPACE 2
UCBCMXT  DS    CL32
UCBDCE   DS    CL256
UCBDCEL  DC    H'256'
DEVNUM   DS    F
DEVCHAR  DS    F
DEVT     DS    F
DEVN     DS    H
DEVNCHAR DS    CL4
DEVNUNPK DS    CL5
DEVNWORK DS    CL3
RCR15    DS    X
         DC    C'RTNCD'
RTNCD    DC    F'Ø'
RSNCD    DC    F'Ø'
RTCD     DC    F'Ø'
```

```
RSCD     DC    F'Ø'
XASYNCH  DC    CL3' '
XDEVN    DC    XL2'ØØØØ'
XQRYSIZE DC    XL2'ØØØØ'
         DC    CL1Ø'XQRYINFO'
XQRYINFO DC    512CL1' '
         DC    C'XRETINFO'
XRETINFO DS    ØF,CL1ØØ
         ORG   XRETINFO
RTC      DC    F'Ø'
RSC      DC    F'Ø'
         ORG   XRETINFO+1ØØ
         DC    CL4'XECB'
XECB     DC    F'Ø'
TEST     DC    256AL1(*-TEST)
TRTAB    DC    256AL1(*-TRTAB)
         ORG   TRTAB+X'FA'
         DC    C'ABCDEF'
         ORG
         ANTRQSTL NAME=P_LIST,BASE=ØF
         LTORG
ZERO     EQU   X'ØØ'
PARMDS   DSECT
ADEVDS   DS    A
ABUFDS   DS    A
AVOLDS   DS    A
DEVDS    DSECT
LENDEV   DS    H
STRTDEV  DS    H
BUFDS    DSECT
BUFLEN   DS    H
BUFFER   DS    CL32ØØØ
VOLDS    DSECT
VOLLEN   DS    H
VOLTAB   DS    CL32ØØØ
         SPACE 2
         IEFUCBOB
         END   IPPFCQRY
```

IPPUCBRX

The last program is a small one. IPPUCBRX obtains a volume serial with an address as input:

```
IPPUCBRX TITLE 'IPPUCBRX - GET VOLSER FOR A CUA ADDRESS'
         SPACE 2
* ——————————————————————————————————————————————————————————————————
* THIS PROGRAM PERFORMS TWO FUNCTIONS :
*  - RETURN VOLSER OF A GIVEN DASD ADDRESS
*  - RETURN VOLSER OF NEXT ONLINE ADDRESS AFTER A GIVEN DASD ADDRESS
* PARAMETERS : THIS PROGRAM IS CALLED FROM REXX USING THE
```

```
*  "LINKMVS" HOST ENVIRONMENT - PARAMS ARE SETUP ACCORDINGLY
*  REMARK : PARAMETERS CAN BE BOTH INPUT AND OUTPUT
* PARAM1 : LENGTH  DC H (INPUT VALUE IS 4)
*          CUA     DC H (INPUT CUA IN BINARY)
* PARAM2 : LENGTH  DC H   (INPUT VALUE IS 1Ø)
*          VOLSER  DC CL6 (INPUT IS BLANK OR "NXTONL")
*          CUA     DC CL4 (OUTPUT ONLY : CUA FOR THIS VOLSER IN EBCDIC)
* ------------------------------------------------------------------------
         SPACE 2
         EJECT
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
R11      EQU   11
R12      EQU   12
R13      EQU   13
R14      EQU   14
R15      EQU   15
         SPACE 2
RDEVDS   EQU   2
RBUFDS   EQU   2
RVOLDS   EQU   2
         SPACE 2
RBUFLEN  EQU   3
RBUFFER  EQU   4
RVOLLEN  EQU   5
RVOLTAB  EQU   6
RUCB     EQU   7
         SPACE 2
RPARM    EQU   8
RET      EQU   9
RW       EQU   1Ø
RW2      EQU   11
         SPACE 2
IPPUCBRX CSECT
         PRINT GEN
         SPACE 2
* ------------------------------------------------------------------------
*        STANDARD INIT ROUTINE
* ------------------------------------------------------------------------
         SPACE 2
INITIAL  DS    ØH
         SAVE  (14,12),,IPPUCBRX.&SYSDATE.&SYSTIME
         LR    R12,R15                LOAD START ADDRESS
         LR    RPARM,R1
```

```
         USING IPPUCBRX,R12          ESTABLISH ADDRESSABILITY
         LA    R5,WORKAREA           ADDRESS WORKAREA
         ST    R5,8(,R13)            STORE FORW POINTER IN CALLING PGM
         ST    R13,4(,R5)            STORE BACKW POINTER IN CALLED(THIS
         LR    R13,R5
         USING WORKAREA,R13
         B     START
         EJECT
* ------------------------------------------------------------------------
*        START PROCESSING
* ------------------------------------------------------------------------
         SPACE 2
START    DS    ØH
         USING PARMDS,RPARM          ACCESS PARAMS PASSED ...
         L     RDEVDS,ADEVDS
         USING DEVDS,RDEVDS
         MVC   DEVN,STRTDEV
         DROP  RDEVDS
         L     RVOLDS,AVOLDS
         USING VOLDS,RVOLDS
         LA    RVOLTAB,VOLTAB
         DROP  RVOLDS
         XC    UCBWAREA,UCBWAREA     CLEAR UCB WORK AREA WITH BIN ZEROS
         XC    UCBAREA,UCBAREA       CLEAR UCB AREA WITH BIN ZEROS
         UCBSCAN COPY,WORKAREA=UCBWAREA,UCBAREA=UCBAREA,RANGE=ALL,     X
               DEVCLASS=DASD,DEVN=DEVN,DEVNCHAR=DEVNCHAR,DYNAMIC=YES,  X
               CMXTAREA=UCBCMXT,DCEAREA=UCBDCE,DCELEN=UCBDCEL
         LA    RUCB,UCBAREA          LOAD ADDRESS UCB RETURNED BY SCAN
         USING UCBOB,RUCB            EST ADDRESSABILITY
         TM    UCBSTAT,UCBONLI       IS DEVICE ONLINE
         BZ    CHKPARM               NO, THEN CHECK PARAMETER
         MVC   Ø(6,RVOLTAB),UCBVOLI  DEVICE WAS ONLINE, SO FILL VOLSER
         MVC   6(4,RVOLTAB),DEVNCHAR  AND DEVICE ADDRESS
         B     GETOUT
CHKPARM  CLC   Ø(6,RVOLTAB),=C'NXTONL'  NEXT ONLINE ADDRESS ASKED
         BNE   GETOUT                NO THEN ERROR
UCBLOOP  LTR   R15,R15                YES, THEN LOOP ...
         BNZ   GETOUT
         UCBSCAN COPY,WORKAREA=UCBWAREA,UCBAREA=UCBAREA,RANGE=ALL,     X
               DEVCLASS=DASD,DEVN=DEVN,DEVNCHAR=DEVNCHAR,DYNAMIC=YES,  X
               CMXTAREA=UCBCMXT,DCEAREA=UCBDCE,DCELEN=UCBDCEL
         LA    RUCB,UCBAREA          LOAD ADDRESS UCB RETURNED BY SCAN
         USING UCBOB,RUCB            EST ADDRESSABILITY
         TM    UCBSTAT,UCBONLI         IS DEVICE ONLINE
         BZ    UCBLOOP               NO THEN GET NEXT UCB ...
         MVC   Ø(6,RVOLTAB),UCBVOLI  FILL VOLSER NEXT ONLINE
         MVC   6(4,RVOLTAB),DEVNCHAR FILL ADDRESS NEXT ONLINE ...
GETOUT   DS    ØH
         EJECT
* ------------------------------------------------------------------------
*        STANDARD RETURN SEQUENCE
* ------------------------------------------------------------------------
         SPACE 2
```

```
RETURN   DS    ØH
         XR    R15,R15
         L     R13,SAVEAREA+4        LOAD ADDRESS WHERE PREV REG STORED
         RETURN (14,12),RC=(15)
         EJECT
* ------------------------------------------------------------------------
*        SAVE AND WORKAREAS
* ------------------------------------------------------------------------
         SPACE 2
WORKAREA DS    ØF
SAVEAREA DS    18F
         SPACE 2
         DS    ØF
         DC    C'UCBW'
UCBWAREA DS    CL1ØØ
         SPACE 2
         DS    ØF
         DC    C'UCBA'
UCBAREA  DS    CL48                  THE UCB AREA
         SPACE 2
         SPACE 2
         DS    ØF
         DC    C'UCBM'
UCBCMXT  DS    CL32
         DS    ØF
         DC    C'UCBD'
UCBDCE   DS    CL256
UCBDCEL  DC    H'256'
         DS    ØF
         DC    C'DEVN'
DEVNUM   DS    F
DEVCHAR  DS    F
DEVT     DS    F
DEVN     DS    H
DEVNCHAR DS    CL4
         LTORG
ZERO     EQU   X'ØØ'
PARMDS   DSECT
ADEVDS   DS    A
AVOLDS   DS    A
DEVDS    DSECT
LENDEV   DS    H
STRTDEV  DS    H
VOLDS    DSECT
VOLLEN   DS    H
VOLTAB   DS    CL1Ø
         SPACE 2
         IEFUCBOB
         END   IPPUCBRX
```

Listing WLM scheduling environments and resources

INTRODUCTION

A WLM scheduling environment is a list of resource names with their required states. Each entry in the list consists of a resource name and a required state of either 'ON' or 'OFF'. The current state of a resource can be 'ON', 'OFF', or 'RESET'. The scheduling environment is 'AVAILABLE' when all resources in the list are set to their required state. Scheduling environments and resource names reside in the WLM service definition and apply across the entire sysplex, but the resource states – and the scheduling environment availability – can be different in each system. The state of a WLM resource is controlled by system command 'F WLM,RESOURCE=resourcenane,ON | OFF | RESET'.

The parameter 'SCHENV=' on the JCL JOB statement associates a job with a scheduling environment. The job will be scheduled on the system that first satisfies all of the resource state requirements of the scheduling environment.

SCHENV	Resource(s)	Required State	Current State
NOMAINT	DAY	ON	ON
	NIGHT	OFF	OFF
	DB2TEST	ON	ON
	CICSTEST	ON	ON
BACKUP	DAY	OFF	OFF
	NIGHT	ON	ON
	DB2TEST	OFF	ON

Figure 1: An example scheduling environment

In Figure 1, scheduling environment 'NOMAINT' is available because all resources in the list have been set to their required state. Jobs specifying 'SCHENV=NOMAINT' on the JCL job statement will be scheduled for execution. Scheduling environment 'BACKUP' is

unavailable because resource DB2TEST does not have its required state. Jobs specifying 'SCHENV=BACKUP' on the JCL job statement will be put in the input queue until all resource state requirements are satisfied.

APPLICATION DESCRIPTION

At our site, several scheduling environments have been defined, reflecting a mix of required resource states. WLM resource states are managed by system automation product CA-OPS/MVS. Programmers can specify a scheduling environment on the JCL job control statement, depending on the resources their job needs. To provide them with an up-to-date list of all scheduling environments and related resources, I wrote two Assembler programs 'WLMSERE' and 'WLMSEUSE' combined with a small REXX program and an ISPF panel. WLMSERE is a TSO/E command processor program which displays the following information in one or more PUTLINE messages:

- OS/390 system name.
- Scheduling environment: name, availability, number of depending jobs.
- Resource: name, required state, current state.

Similar data can be obtained using SDSF commands 'SE' and 'RES'. However, WLMSERE combines both SDSF functions in a single view. Since messages are generated by TSO/E PUTLINE, output can be trapped in REXX variables and used for further processing (eg system automation). If the program is called within a CA-OPS/MVS OSF server, output is available in the REXX external data queue. Some sample program output is shown below:

```
Command ==> TSO WLMSERE SYSTEM(MVSTEST) SCHENV(DAY)

System     Schenv     Avail      Schenv     Resource   Required   Current
                                 Users      Name       State      State
MVSTEST    DAY        N          2          DAY        ON         ON
MVSTEST    DAY        N          2          NIGHT      OFF        OFF
MVSTEST    DAY        N          2          DB2TEST    ON         RESET
```

In the above example, scheduling environment 'DAY' is not available even though two active jobs are still depending on it. This arises when one or more resource states have been changed by the operator (or

system automation) after the jobs were scheduled for execution.

ASSEMBLER PROGRAM WLMSERE

WLMSERE accepts two keyword operands: SCHENV() and SYSTEM(). SCHENV() specifies the WLM scheduling environment name to be found. Blank or '*' forces all entries to be listed. SYSTEM() specifies the system name of an OS/390 image in the Sysplex. Specify '*' to scan all images. If left blank, only the current system's entries are displayed. TSO/E Service Routine IKJPARS is called to validate the command operands passed by the caller.

IBM Scheduling Environments Query Service 'IWMSEQRY' provides information about the scheduling environments and their resources on all systems in the sysplex. Output is returned in an answer area mapped by IWMSET macro (see the *MVS Data Areas* book). Supervisor state or program key mask allowing keys 0-7 is required. This means you need to include your authorization SVC (eg SVC 235) or update parmlib member IKJTSO*xx* to define authorized commands. The module must be linked with authorization code 1 (AC=1) and reside in an APF authorized load library.

ASSEMBLER PROGRAM WLMSEUSE

WLMSEUSE is called by WLMSERE to find the number of active jobs relying on a scheduling environment. Scheduling environment data is obtained from the Subsystem Job Block (SJB) associated with each address space.

On return, WLMSEUSE stores the number of active jobs in GPR1. GPR15 contains the following return codes:

- 0 – SCHENV is not used
- 4 – SCHENV is inuse by at least one active job
- 8 – Error.

WLMSEUSE uses JES2 mapping macros. You need to include JES2 SHASMAC and SHASSRC libraries on the SYSLIB DD JCL statement to compile the source code.

REXX PROGRAM WLMSERE

WLMSERE is a TSO/E REXX program to process output generated by load module WLMSERE. Output messages are trapped in REXX stem variables and stored in an ISPF table. ISPF TBDISPL service is used to display the results in a scrollable panel.

Sample LinkEdit JCL statements:

```
//LKED        EXEC PGM=HEWLHØ96,PARM='XREF,LET,LIST,NCAL'
//OBJECT  DD DISP=SHR,DSN=your.object.library
//SYSLMOD DD DISP=SHR,DSN=your.APF.authorized.load.library
//SYSPRINT    DD SYSOUT=*
//SYSLIN  DD *
  SETCODE AC(1)
  INCLUDE OBJECT(WLMSERE)
  INCLUDE OBJECT(WLMSEUSE)
  NAME WLMSERE(R)
```

Sample IKJTSO*xx* parmlib statements:

```
AUTHCMD NAMES(     /* AUTHORIZED COMMANDS                          */ +
AD       ADDSD     /* RACF COMMANDS                               */ +
IEBCOPY            /*                                             */ +
RECEIVE            /* TSO COMMANDS                                */ +
WLMSERE            /* LIST WLM SCHED. ENV.                        */ )
```

Issue the TSO command ‘PARMLIB UPDATE(xx)’ to dynamically update the TSO/E system values. This step can be skipped if you use your authorization SVC. All programs have been tested on OS/390 Version 2 Release 5 and Version 2 Release 8 with the corresponding versions of JES2.

WLMSERE

```
WLMSERE  CSECT
WLMSERE  AMODE 31
WLMSERE  RMODE ANY
*-------------------------------------------------------------------*
*        FUNCTION : LIST WLM SCHEDULING ENVIRONMENTS AND RESOURCES  *
*     ENVIRONMENT : TSO/E COMMAND PROCESSOR                         *
*       ARGUMENTS : - SCHENV(......)                                *
*                       SCHENV NAME                                 *
*                       BLANK OR '*' FOR ALL SCHEDULING ENVIRONMENTS*
*                   - SYSTEM(......)                                *
*                       OS/39Ø SYSTEM NAME                          *
*                       '*' FOR ALL SYSTEMS                         *
*                       BLANK FOR CURRENT SYSTEM                    *
```

```
*          OUTPUT : TSO/E PUTLINE MESSAGE(S) DISPLAYING               *
*                   SCHENV/RESOURCE RELATIONSHIP                      *
*----------------------------------------------------------------------*
         YREGS                          REGISTER EQUATES
*----------------------------------------------------------------------*
         BAKR  R14,Ø                    SAVE STATUS ON LINKAGE STACK
         USING WLMSERE,R12              R12 IS BASE FOR ASSEMBLY
         LR    R12,R15                  R12 IS BASE FOR EXECUTION
*----------------------------------------------------------------------*
*        PARSE COMMAND INPUT                                           *
*----------------------------------------------------------------------*
         LR    R2,R1                    SAVE CPPL ADDRESS (COMMAND
*                                       PROCESSOR PARAMETER LIST)
         USING CPPL,R2                  CPPL ADDRESSABILITY
         LA    R3,PPLAREA               ADDR(PARSE PARAMETER LIST)
         USING PPL,R3                   PPL ADDRESSABILITY
         LA    R4,IOPLADR               ADDR(I/O PARM LIST)
         USING IOPL,R4                  IOPL ADDRESSABILITY
*----------------------------------------------------------------------*
*        INITIALIZE PARSE PARAMETER LIST                               *
*        CALL IKJPARS SERVICE ROUTINE                                  *
*----------------------------------------------------------------------*
         L     R1,CPPLUPT               ADDR(USER PROFILE TABLE)
         ST    R1,PPLUPT                STORE IN PPL
         ST    R1,IOPLUPT               STORE IN IOPL
         L     R1,CPPLECT               ADDR(ENVIRONMENT CONTROL TABLE)
         ST    R1,PPLECT                STORE IN PPL
         ST    R1,IOPLECT               STORE IN IOPL
         LA    R1,ECB                   ADDR(EVENT CONTROL BLOCK)
         ST    R1,PPLECB                STORE IN PPL
         ST    R1,IOPLECB               STORE IN IOPL
         L     R1,APCL                  ADDR(PARAMETER CONTROL LIST)
         ST    R1,PPLPCL                STORE IN PPL
         LA    R1,PPLPDL                ADDR(PARAMETER DESCRIPTOR LIST)
         ST    R1,PPLANS                STORE IN PPL
         L     R1,CPPLCBUF              ADDR(COMMAND BUFFER)
         ST    R1,PPLCBUF               STORE IN PPL
         XC    PPLUWA,PPLUWA            CLEAR WORK AREA
         XC    ECB,ECB                  CLEAR EVENT CONTROL BLOCK
         LR    R1,R3                    LOAD ADDR(PPL)
         CALLTSSR EP=IKJPARS            CALL IKJPARS
         LTR   R15,R15                  CHECK RETURN CODE
         BNZ   RETURN                   RC ¬= Ø ; RETURN
         DROP  R2,R3,R4
*----------------------------------------------------------------------*
*        GET CURRENT SYSTEM NAME                                       *
*----------------------------------------------------------------------*
         L     R1,CVTPTR                ADDR(CVT)
         USING CVTMAP,R1                CVT ADDRESSABILITY
         MVC   CURRENT_SYSTEM,CVTSNAME  SAVE CURRENT SYSTEM NAME
         DROP  R1
```

```
*--------------------------------------------------------------------*
*        PROCESS PARSE RESULTS                                       *
*--------------------------------------------------------------------*
ARG#SCH  DS    ØH
         MVI   SCHENV_NAME,C' '        CLEAR SCHENV_NAME AREA
         MVC   SCHENV_NAME+1(L'SCHENV_NAME-1),SCHENV_NAME REPEAT
         L     R2,PPLPDL               ADDR(PARMS DESCRIPTOR LIST)
         USING IKJPARMD,R2             PDL ADDRESSABILITY
         L     R4,ARGSCH               ADDR(SCHENV NAME ARGUMENT)
         LH    R1,ARGSCH+4             LENGTH(SCHENV NAME ARGUMENT)
         LTR   R1,R1                   SCHENV NAME SUPPLIED?
         BZ    ARG#SYS                 NO, SKIP
         BCTR  R1,Ø                    LENGTH = LENGTH - 1
         EX    R1,GETSCH               GET SCHENV NAME ARGUMENT
ARG#SYS  DS    ØH
         MVC   SYSTEM_NAME,CURRENT_SYSTEM DEFAULT = CURRENT SYSTEM NAME
         L     R4,ARGSYS               ADDR(SYSTEM NAME ARGUMENT)
         LH    R1,ARGSYS+4             LENGTH(SYSTEM NAME ARGUMENT)
         LTR   R1,R1                   SYSTEM NAME SUPPLIED?
         BZ    GETSTOR                 NO, SKIP
         MVI   SYSTEM_NAME,C' '        CLEAR SYSTEM_NAME AREA
         MVC   SYSTEM_NAME+1(L'SYSTEM_NAME-1),SYSTEM_NAME REPEAT
         BCTR  R1,Ø                    LENGTH = LENGTH - 1
         EX    R1,GETSYS               GET SYSTEM NAME ARGUMENT
         B     GETSTOR                 CONTINUE
         DROP  R2
*--------------------------------------------------------------------*
*        EXECUTED INSTRUCTIONS                                       *
*--------------------------------------------------------------------*
GETSCH   MVC   SCHENV_NAME(Ø),Ø(R4)    SAVE SCHENV NAME ARGUMENT
GETSYS   MVC   SYSTEM_NAME(Ø),Ø(R4)    SAVE SYSTEM NAME ARGUMENT
*--------------------------------------------------------------------*
*        OBTAIN STORAGE FOR IWMSEQRY ANSWER AREA                     *
*--------------------------------------------------------------------*
GETSTOR  DS    ØH
         L     R1,ANSLEN               ANSWER AREA SIZE
         STORAGE OBTAIN,LENGTH=(R1),ADDR=(R2),LOC=ANY
         ST    R2,AANSAREA             SAVE ADDR(ANSWER AREA)
*--------------------------------------------------------------------*
*        CALL IWMSEQRY                                               *
*--------------------------------------------------------------------*
*        LA    R1,1                    Insert your SVC code
*        SVC   235                     to get authorized
         MODESET MODE=SUP,KEY=ZERO     SUPERVISOR STATE ; KEY ZERO
         IWMSEQRY ANSAREA=(R2),        ADDR(ANSWER AREA)               X
               ANSLEN=ANSLEN,          ANSWER AREA SIZE                X
               QUERYLEN=RQDLEN,        REQUIRED AREA SIZE              X
               RETCODE=RETCODE,        RETURN CODE                     X
               RSNCODE=RSNCODE         REASON CODE
         USING SETHDR,R2               HEADER ADDRESSABILITY
         MODESET MODE=PROB,KEY=NZERO   PROBLEM STATE ; KEY NZERO
```

```
*        LA    R1,Ø                    Insert your SVC code
*        SVC   235                     to get unauthorized
*-------------------------------------------------------------------*
*        CHECK RETURN/REASON CODES                                  *
*-------------------------------------------------------------------*
         L     R15,RETCODE             LOAD IWMSEQRY RETURN CODE
         C     R15,=A(IWMRETCODEOK)    IWMSEQRY CALL SUCCESSFUL?
         BE    PROCRSLT                YES, PROCESS RESULTS
         L     R15,RSNCODE             NO, LOAD IWMSEQRY REASON CODE
         C     R15,=A(IWMRSNCODEOUTPUTAREATOOSMALL)
*                                      ANSWER AREA TOO SMALL?
         BNE   RELSTOR                 NO, RELEASE AREA AND RETURN
*-------------------------------------------------------------------*
*        OBTAIN A NEW ANSWER AREA                                   *
*-------------------------------------------------------------------*
         L     R1,ANSLEN               CURRENT ANSWER AREA SIZE
         STORAGE RELEASE,LENGTH=(R1),ADDR=(R2) RELEASE STORAGE
         L     R1,RQDLEN               GET REQUIRED AREA SIZE
         ST    R1,ANSLEN               SET REQUIRED SIZE
         B     GETSTOR                 AND OBTAIN A NEW ANSWER AREA
*-------------------------------------------------------------------*
*        PROCESS OUTPUT RETURNED BY IWMSEQRY                        *
*-------------------------------------------------------------------*
PROCRSLT DS    ØH
*-------------------------------------------------------------------*
*        LOCATE SYSTEM SECTION (SETSYS)                             *
*-------------------------------------------------------------------*
         L     R11,SET_OFFSET_SYS      OFFSET TO SETSYS
         AR    R11,R2                  ADD OFFSET TO HEADER
         USING SETSYS,R11              SETSYS ADDRESSABILITY
         LH    R1Ø,SET_NUMBER_SYS      NUMBER OF SETSYS ENTRIES
*-------------------------------------------------------------------*
*        LOCATE SYSTEM STATUS HEADER (SETSYH)                       *
*        PROCESS SYSTEM ENTRIES (LOOP1)                             *
*-------------------------------------------------------------------*
SYS#LOOP DS    ØH
         CLI   SYSTEM_NAME,C'*'        LIST ALL SYSTEMS?
         BE    SYHPROC                 YES, CONTINUE
         CLC   SET_SYS_NAME,SYSTEM_NAME REQUIRED SYSTEM?
         BNE   SYS#NEXT                NO, TRY NEXT SYSTEM ENTRY
SYHPROC  DS    ØH
         L     R3,SET_SYS_STATUS_PTR   SYSTEM STATUS POINTER (SETSYH)
         LTR   R3,R3                   CHECK ADDRESS
         BZ    SYS#NEXT                ZERO, TRY NEXT SYSTEM ENTRY
         USING SETSYH,R3               SETSYH ADDRESSABILITY
*-------------------------------------------------------------------*
*        LOCATE SCHEDULING ENVIRONMENT SYSTEM STATUS SECTION (SETSES) *
*-------------------------------------------------------------------*
         L     R9,SET_OFFSET_SES       OFFSET OF SCHED. ENVIRONMENT
*                                      STATUS SECTION (SETSES)
         AR    R9,R3                   ADDR(FIRST SETSES ENTRY)
```

```
         USING SETSES,R9               SETSES ADDRESSABILITY
*--------------------------------------------------------------------*
*        LOCATE RESOURCE SYSTEM STATUS SECTION (SETRES)              *
*--------------------------------------------------------------------*
         L     R1,SET_OFFSET_RES       OFFSET OF RESOURCE STATUS
*                                      SECTION (SETRES)
         AR    R1,R3                   ADDR(FIRST SETRES ENTRY)
         ST    R1,ASETRES              SAVE ADDR(FIRST SETRES)
         DROP  R3
*--------------------------------------------------------------------*
*        LOCATE SCHEDULING ENVIRONMENT SECTION (SETSE)               *
*--------------------------------------------------------------------*
         L     R7,SET_OFFSET_SE        OFFSET OF SCHENV SECTION (SE)
         LH    R8,SET_NUMBER_SE        NUMBER OF SCHENV ENTRIES
         AR    R7,R2                   ADD OFFSET TO HEADER
         USING SETSE,R7                SE SECTION ADDRESSABILITY
*--------------------------------------------------------------------*
*        PROCESS SCHEDULING ENVIRONMENT ENTRIES (LOOP2)              *
*--------------------------------------------------------------------*
SE#LOOP  DS    ØH
         CLI   SCHENV_NAME,C' '        LIST ALL SCHENV?
         BE    SE#STAT                 YES, GET SCHENV STATUS
         CLI   SCHENV_NAME,C'*'        LIST ALL SCHENV?
         BE    SE#STAT                 YES, GET SCHENV STATUS
         CLC   SET_SE_SCHENV_NAME,SCHENV_NAME   SCHENV TO BE FOUND?
         BNE   SE#NEXT                 NO, TRY NEXT SCHENV
*--------------------------------------------------------------------*
SE#STAT  DS    ØH
         MVI   MSGTEXT,C' '            CLEAR MESSAGE AREA
         MVC   MSGTEXT+1(MSGTEXTL-1),MSGTEXT REPEAT
         MVC   SE_SYSTEM,SET_SYS_NAME       GET SYSTEM NAME
         MVC   SE_NAME,SET_SE_SCHENV_NAME   GET SCHENV NAME
         MVI   SE_AVL,C'Y'                  SET STATUS = AVAILABLE
         CLI   SET_SES_STATUS,SET_SES_AVAILABLE  STATUS = AVAILABLE?
         BE    SE#USE                       YES, CONTINUE
         MVI   SE_AVL,C'N'                  SET STATUS = NOT AVAILABLE
*--------------------------------------------------------------------*
*        CHECK IF SCHEDULING ENVIRONMENT IS USED BY ACTIVE JOBS      *
*        ON CURRENT SYSTEM. CALL WLMSEUSE.                           *
*--------------------------------------------------------------------*
SE#USE   DS    ØH
         MVI   SE_USECOUNT+L'SE_USECOUNT-1,C'?'   SET SE_USECOUNT = ?
         CLC   SET_SYS_NAME,CURRENT_SYSTEM        CURRENT SYSTEM?
         BNE   SE#SR                              NO, SKIP
         MVI   SE_USECOUNT+L'SE_USECOUNT-1,C'Ø'   SET SE_USECOUNT = Ø
         MVC   WLMSEUSE_SCHENV,SET_SE_SCHENV_NAME SUPPLY SCHENV NAME
         LA    R1,WLMSEUSE_PARM        ADDR(SCHENV POINTER)
         L     R15,=V(WLMSEUSE)        ADDR(WLMSEUSE ROUTINE)
         BASR  R14,R15                 CALL WLMSEUSE
         B     *+4(R15)                BRANCH ON RETURN CODE
         B     SE#SR                   RC = Ø  ; SCHENV IS NOT INUSE
```

```
         B     SE#INUSE                RC = 4  ; SCHENV IS INUSE
         B     SE#SR                   RC = 8  ; ERROR
SE#INUSE DS    ØH
         CVD   R1,DOUBLE               CONVERT USE COUNT TO DECIMAL
         MVC   WORK,=X'4Ø2Ø2Ø2Ø2Ø2Ø2Ø2Ø'    INSERT EDIT PATTERN
         ED    WORK,DOUBLE+4           CONVERT TO PRINTABLE TEXT
         MVC   SE_USECOUNT,WORK+4      GET USE COUNT
*--------------------------------------------------------------------*
*        LOCATE SCHENV/RESOURCE RELATIONSHIP SECTION (SETSR)         *
*--------------------------------------------------------------------*
SE#SR    DS    ØH
         L     R5,SET_SE_SR_OFFSET     OFFSET OF FIRST SCHENV/RESOURCE
*                                       (SR) RELATIONSHIP ENTRY
*                                       FOR THIS SCHED. ENVIRONMENT
         L     R6,SET_SE_SR_COUNT      NUMBER OF SR ENTRIES
         LTR   R6,R6                   CHECK NUMBER
         BZ    DISPLAY                 NO RESOURCE RELATIONSHIPS
         AR    R5,R2                   ADD OFFSET TO HEADER
         USING SETSR,R5                SR RELATIONSHIP ADDRESSABILITY
*--------------------------------------------------------------------*
*        PROCESS RESOURCE ENTRIES (LOOP3)                            *
*--------------------------------------------------------------------*
SR#LOOP  DS    ØH
         L     R1,SET_SR_RE_OFFSET     OFFSET OF RESOURCE ENTRY (RE)
*                                       FROM BEGINNING OF THE SET
         AR    R1,R2                   ADD OFFSET TO HEADER
         USING SETRE,R1                RE ADDRESSABILITY
         MVC   RE_NAME,SET_RE_RESOURCE_NAME GET RESOURCE NAME
         DROP  R1
*--------------------------------------------------------------------*
*        GET REQUIRED STATE OF THE RESOURCE FOR THE                  *
*        SCHEDULING ENVIRONMENT TO BE AVAILABLE                      *
*--------------------------------------------------------------------*
         MVC   RE_REQSTAT,=CL3'ON'     SET REQUIRED STATE = ON
         TM    SET_SR_RESOURCE_STATE,SET_SR_ON   REQUIRED ON?
         BO    SR#RES                  YES, CONTINUE
         MVC   RE_REQSTAT,=CL3'OFF'    SET REQUIRED STATE = OFF
         TM    SET_SR_RESOURCE_STATE,SET_SR_OFF  REQUIRED OFF?
         BO    SR#RES                  YES, CONTINUE
         MVI   RE_REQSTAT,C'?'         NO, REQUIRED STATE IS UNKNOWN
*--------------------------------------------------------------------*
*        LOCATE RESOURCE SYSTEM STATUS SECTION (SETRES)              *
*--------------------------------------------------------------------*
SR#RES   DS    ØH
         L     R1,SET_SR_RE_INDEX      GET INDEX OF THE RESOURCE ENTRY
         BCTR  R1,Ø                    INDEX = INDEX - 1
         M     RØ,=A(SETRES_LEN)       R1 = OFFSET WITHIN SETRES
         L     R3,ASETRES              ADDR(FIRST SETRES ENTRY)
         AR    R3,R1                   ADD OFFSET
         USING SETRES,R3               SETRES ADDRESSABILITY
         MVC   RE_CURSTAT,=CL5'ON'          SET CURRENT STATUS = ON
         CLI   SET_RES_STATE,SET_RES_ON     STATUS = ON?
```

```
         BE    DISPLAY                      YES, CONTINUE
         MVC   RE_CURSTAT,=CL5'OFF'         SET CURRENT STATUS = OFF
         CLI   SET_RES_STATE,SET_RES_OFF    STATUS = OFF?
         BE    DISPLAY                      YES, CONTINUE
         MVC   RE_CURSTAT,=CL5'RESET'       SET CURRENT STATUS = RESET
         CLI   SET_RES_STATE,SET_RES_RESET  STATUS = RESET?
         BE    DISPLAY                      YES, CONTINUE
         MVI   RE_CURSTAT,C'?'              NO, CURRENT STATUS UNKNOWN
         DROP  R3
*-------------------------------------------------------------------*
*        DISPLAY RESULTS                                            *
*-------------------------------------------------------------------*
DISPLAY  DS    ØH
         PUTLINE PARM=PUTBLOCK,        ISSUE PUTLINE MESSAGE          X
               OUTPUT=(MESSAGE,TERM,SINGLE,DATA),                     X
               MF=(E,IOPLADR)
*-------------------------------------------------------------------*
*        END OF LOOP3 (RESOURCE ENTRIES)                            *
*-------------------------------------------------------------------*
SR#NEXT  DS    ØH
         LTR   R6,R6                  ANY SR ENTRY LEFT?
         BZ    SE#NEXT                NO, ESCAPE
         LA    R5,SETSR_LEN(,R5)      ADDR(NEXT SR ENTRY)
         BCT   R6,SR#LOOP             PROCESS NEXT SR ENTRY
*-------------------------------------------------------------------*
*        END OF LOOP2 (SCHENV ENTRIES)                              *
*-------------------------------------------------------------------*
SE#NEXT  DS    ØH
         LA    R7,SETSE_LEN(,R7)      ADDR(NEXT SETSE ENTRY)
         LA    R9,SETSES_LEN(,R9)     ADDR(NEXT SETSES ENTRY)
         BCT   R8,SE#LOOP             PROCESS NEXT SE/SES ENTRIES
*-------------------------------------------------------------------*
*        END OF LOOP1 (SYSTEM ENTRIES)                              *
*-------------------------------------------------------------------*
SYS#NEXT DS    ØH
         LA    R11,SETSYS_LEN(,R11)   ADDR(NEXT SETSYS ENTRY)
         BCT   RIØ,SYS#LOOP           PROCESS NEXT SETSYS ENTRY
*-------------------------------------------------------------------*
*        RELEASE VIRTUAL STORAGE                                    *
*        RETURN                                                     *
*-------------------------------------------------------------------*
RELSTOR  DS    ØH
         L     R1,ANSLEN              LENGTH(ANSWER AREA)
         L     R2,AANSAREA            ADDR(ANSWER AREA)
         STORAGE RELEASE,LENGTH=(R1),ADDR=(R2)   RELEASE ANSWER AREA
         L     R15,RETCODE            LOAD IWMSEQRY RETURN CODE
RETURN   DS    ØH
         PR                           RESTORE REGISTERS 2-14 FROM
*                                     LINKAGE-STACK AND RETURN
         DROP  R2,R5,R7,R9,R11
*-------------------------------------------------------------------*
```

```
*        CONSTANTS AND VARIABLES                                        *
*-----------------------------------------------------------------------*
WORK           DS XL8                  WORK AREA
DOUBLE         DS D                    DOUBLE WORD
SCHENV_NAME    DS CL16                 SCHED. ENVIRONMENT ARGUMENT
SYSTEM_NAME    DS CL8                  SYSTEM NAME ARGUMENT
CURRENT_SYSTEM DS CL8                  CURRENT SYSTEM NAME
*
AANSAREA DS    F                       ADDR(ANSWER AREA)
ANSLEN   DC    A(50*1024)              INITIAL LEN(ANSWER AREA) = 50K
RQDLEN   DS    F                       REQUIRED ANSWER AREA SIZE
RETCODE  DS    F                       RETURN CODE
RSNCODE  DS    F                       REASON CODE
*
ASETRES  DS    F                       ADDR(FIRST SETRES ENTRY)
*-----------------------------------------------------------------------*
*        WLMSEUSE PARMS                                                 *
*-----------------------------------------------------------------------*
WLMSEUSE_PARM    DC A(WLMSEUSE_SCHENVL)  ADDR(SCHENV AREA)
WLMSEUSE_SCHENVL DC Y(L'WLMSEUSE_SCHENV) LENGTH(WLMSEUSE_SCHENV)
WLMSEUSE_SCHENV  DS CL16                 SCHENV TO BE FOUND
*-----------------------------------------------------------------------*
*        IKJPARS PARAMETERS                                             *
*-----------------------------------------------------------------------*
ECB      DS    F                       EVENT CONTROL BLOCK
PPLPDL   DS    F                       ADDR(PARAMETER DESCRIPTOR LIST)
PPLAREA  DS    CL(PPLLEN)              PARSE PARAMETER LIST AREA
APCL     DC    A(PARMCNTL)             ADDR(PARAMETER CONTROL LIST)
*-----------------------------------------------------------------------*
*        PARSE PARAMETER CONTROL LIST (PCL)                             *
*-----------------------------------------------------------------------*
PARMCNTL IKJPARM
SCHKEYW  IKJKEYWD DEFAULT='SCHENV'
         IKJNAME 'SCHENV',SUBFLD=SCHENV,ALIAS=('SCH')
SYSKEYW  IKJKEYWD DEFAULT='SYSTEM'
         IKJNAME 'SYSTEM',SUBFLD=SYSTEM,ALIAS=('SYS')
SCHENV   IKJSUBF
ARGSCH   IKJIDENT 'SCHENV',MAXLNTH=16,CHAR,UPPERCASE,                   X
               HELP='WLM SCHENV NAME ; MAX 16 CHARACTERS'
SYSTEM   IKJSUBF
ARGSYS   IKJIDENT 'SYSTEM',MAXLNTH=8,CHAR,UPPERCASE,                    X
               HELP='MVS SYSTEM NAME ; MAX 8 CHARACTERS'
         IKJENDP
*-----------------------------------------------------------------------*
*        PUTLINE MACRO CONTROL BLOCK                                    *
*-----------------------------------------------------------------------*
PUTBLOCK     PUTLINE MF=L              PUTLINE LIST FORMAT
MESSAGE      DC  Y(MSGTEXTL+4)         LENGTH(MESSAGE TEXT + HEADER)
             DC  Y(0)                  RESERVED
MSGTEXT      EQU *                     MESSAGE TEXT
SE_SYSTEM    DS  CL8                   SYSTEM NAME
```

```
            DS  CL1
SE_NAME     DS  CL16                 SCHED. ENVIRONMENT NAME
            DS  CL1
SE_AVL      DS  CL1                  SCHENV AVAILABLE (Y/N)
            DS  CL1
SE_USECOUNT DS  CL4                  SCHENV USE COUNT
            DS  CL1
RE_NAME     DS  CL16                 RESOURCE NAME
            DS  CL1
RE_REQSTAT  DS  CL3                  RESOURCE REQUIRED STATE
            DS  CL1
RE_CURSTAT  DS  CL5                  RESOURCE CURRENT STATE
MSGTEXTL    EQU *-MSGTEXT            LENGTH(MESSAGE TEXT)
IOPLADR     DC  4F'Ø'                SPACE FOR I/O PARM LIST
*-------------------------------------------------------------------*
            LTORG
*-------------------------------------------------------------------*
*        MAPPING MACRO'S                                            *
*-------------------------------------------------------------------*
         CVT   DSECT=YES
         IWMYCON
         IWMSET
         IKJCPPL
         IKJIOPL
         IKJPPL
PPLLEN   EQU   *-PPL                  LENGTH(PARSE PARAMETER LIST)
*-------------------------------------------------------------------*
         END
```

WLMSEUSE

```
WLMSEUSE CSECT
WLMSEUSE AMODE 31
WLMSEUSE RMODE ANY
*-------------------------------------------------------------------*
*        FIND NUMBER OF JOBS USING A WLM SCHEDULING ENVIRONMENT     *
*        INPUT :  R1 -> +--------+                                  *
*                       +APARMLST+ -> +-----+----------------------+ *
*                       +--------+    + LEN + SCHENV               + *
*                                     +-----+----------------------+ *
*                                                                   *
*        OUTPUT : R1      -> NUMBER OF JOBS USING SCHENV            *
*                 R15 = Ø -> SCHENV IS NOT INUSE                    *
*                 R15 = 4 -> SCHENV INUSE                           *
*                 R15 = 8 -> ERROR                                  *
*-------------------------------------------------------------------*
         $HASPEQU                    JES2 EQUATES
*-------------------------------------------------------------------*
         BAKR  R14,Ø                 SAVE STATUS ON LINKAGE STACK
         USING WLMSEUSE,R12          R12 IS BASE FOR ASSEMBLY
         LR    R12,R15               R12 IS BASE FOR EXECUTION
```

```
*--------------------------------------------------------------------*
*        GET SCHEDULING ENVIRONMENT SUPPLIED BY CALLER               *
*--------------------------------------------------------------------*
         L     R2,Ø(,R1)               ADDR(PARM LIST)
         LH    R1,Ø(,R2)               LENGTH(PARAMETER)
         LTR   R1,R1                   CHECK LENGTH
         BZ    ERROR                   LENGTH = Ø ; RETURN
         BM    ERROR                   LENGTH < Ø ; RETURN
         C     R1,=A(L'SCHENV)         CHECK LENGTH
         BH    ERROR                   LENGTH > 16 ; RETURN
*
         MVI   SCHENV,C' '             CLEAR SCHENV AREA
         MVC   SCHENV+1(L'SCHENV-1),SCHENV   REPEAT
         BCTR  R1,Ø                    LENGTH = LENGTH - 1
         EX    R1,MOVE#SE              GET SCHEDULING ENVIRONMENT
*--------------------------------------------------------------------*
*        LOCATE ADDRESS SPACE VECTOR TABLE ENTRIES                   *
*--------------------------------------------------------------------*
         L     R1,CVTPTR               ADDR(CVT)
         USING CVT,R1                  CVT ADDRESSABILITY
         L     R2,CVTASVT              ADDR(ADDRESS SPACE VECTOR TABLE)
         USING ASVT,R2                 ASVT ADDRESSABILTY
         LH    R3,ASVTMAXU+2           MAXIMUM NUMBER OF ADDRESS SPACES
         LA    R2,ASVTENTY             ADDR(ASVT ENTRY TABLE)
*--------------------------------------------------------------------*
*        LOCATE JES2 ADDRESS SPACE VECTOR TABLE                      *
*--------------------------------------------------------------------*
         L     R1,CVTJESCT             ADDR(JOB ENTRY SUBSYSTEM
*                                           COMMUNICATION TABLE)
         USING JESCT,R1                JESCT ADDRESSABILTY
         L     R1,JESSSCT              ADDR(FIRST SUBSYSTEM
*                                           COMMUNICATION TABLE)
         USING SSCT,R1                 SSCT ADDRESSABILITY
         L     R1,SSCTSUS2             ADDR(HASP COMMON-STORAGE
*                                           COMMUNICATION TABLE)
         USING HCCT,R1                 HCCT ADDRESSABILITY
         L     R4,CCTHAVT              ADDR(JES2 ADDRESS SPACE
*                                           VECTOR TABLE)
         DROP  R1
*--------------------------------------------------------------------*
*        LOOP THROUGH ADDRESS SPACE VECTOR TABLE                     *
*--------------------------------------------------------------------*
         XR    R7,R7                   CLEAR COUNTER
ASLOOP   DS    ØH
         L     R5,Ø(,R2)               ADDR(ASCB)
         LTR   R5,R5                   ADDRESS SPACE IS ASSIGNED?
         BM    NEXT                    NO, TRY NEXT ADDRESS SPACE
         USING ASCB,R5                 ASCB ADDRESSABILITY
         L     R1,ASCBJBNI             ADDR(JOBNAME FOR JOBS)
         LTR   R1,R1                   JOBNAME AVAILABLE?
         BZ    NEXT                    NO, TRY NEXT ADDRESS SPACE
```

```
*--------------------------------------------------------------------*
*        LOOP THROUGH JES2 ADDRESS SPACE VECTOR TABLE                *
*        GET SCHEDULING ENVIRONMENT FROM SJB (SUBSYSTEM JOB BLOCK)   *
*--------------------------------------------------------------------*
         LH    R1,ASCBASID             GET ADDRESS SPACE ID
         SLL   R1,2                    ASID * 4
         L     R1,Ø(R1,R4)             ADDR(HASP ADDRESS SPACE BLOCK)
         LTR   R1,R1                   HASB AVAILABLE FOR ASID?
         BZ    NEXT                    NO, TRY NEXT ADDRESS SPACE
         USING HASB,R1                 HASB ADDRESSABILITY
         L     R6,HSBSJB               GET FIRST SUBSYSTEM JOB BLOCK
         LTR   R6,R6                   SJB AVAILABLE FOR ASID?
         BZ    NEXT                    NO, TRY NEXT ADDRESS SPACE
         USING SJB,R6                  SJB ADDRESSABILITY
SJBLOOP  DS    ØH
         ICM   R1,B'1111',SJBSJB       LAST SJB?
         BZ    SJBLAST                 YES, EXIT SJB LOOP
         LR    R6,R1                   NO, POINT TO NEXT SJB
         B     SJBLOOP                 REPEAT SJB SCAN
SJBLAST  DS    ØH
         CLC   SCHENV,SJBSCENV         SCHEDULING ENVIRONMENT FOUND?
         BNE   NEXT                    NO, TRY NEXT ADDRESS SPACE
         LA    R7,1(,R7)               YES, COUNTER = COUNTER + 1
*--------------------------------------------------------------------*
NEXT     DS    ØH
         LA    R2,L'ASVTENTY(,R2)      POINT TO NEXT TABEL ENTRY
         BCT   R3,ASLOOP               REPEAT SCAN
*--------------------------------------------------------------------*
         LTR   R1,R7                   CHECK COUNTER
         BZ    NOTUSED                 COUNTER = Ø ; SCHENV NOT USED
USED     DS    ØH
         LA    R15,4                   SET RC = 4 (SCHENV INUSE)
         B     RETURN                  AND RETURN
NOTUSED  DS    ØH
         LA    R15,Ø                   SET RC = Ø (SCHENV NOT USED)
         B     RETURN                  AND RETURN
ERROR    DS    ØH
         XR    R1,R1                   CLEAR COUNTER
         LA    R15,8                   SET RC = 8 (ERROR)
*--------------------------------------------------------------------*
*        RETURN                                                      *
*--------------------------------------------------------------------*
RETURN   DS    ØH
         PR                            RESTORE REGISTERS 2-14 FROM
*                                      LINKAGE-STACK AND RETURN
         DROP  R1,R2,R5,R6
*--------------------------------------------------------------------*
*        EXECUTED INSTRUCTIONS                                       *
*--------------------------------------------------------------------*
MOVE#SE  MVC   SCHENV(Ø),2(R2)         GET SCHEDULING ENVIRONMENT
*--------------------------------------------------------------------*
```

```
*        CONSTANTS AND VARIABLES                                        *
*-----------------------------------------------------------------------*
SCHENV   DS    CL16                    WLM SCHEDULING ENVIRONMENT
*-----------------------------------------------------------------------*
*        MAPPING MACRO'S                                                *
*-----------------------------------------------------------------------*
         CVT   DSECT=YES,PREFIX=NO
         IEECHAIN
         IEFJESCT
         IEFJSCVT
         IHAASCB
         IHAASVT
         $HASB
         $HCCT
         $HFAME
         $SCAT
         $SJB
         $TQE
         $XECB
*-----------------------------------------------------------------------*
         END
```

WLMSERE

```
/* REXX WLMSERE */

 system  = ''
 schenv  = ''
 retcode = Ø
 offset  = Ø

 Do While retcode = Ø
    Call Outtrap('record.')
   "WLMSERE SYS("system") SCH("schenv")"
    Call Outtrap('OFF')

    sys#prev = ''
    se#prev  = ''

   "ISPEXEC CONTROL ERRORS RETURN"
   "ISPEXEC TBCREATE WLMSERE
    NAMES(SYS#NAME SE#NAME SE#AVL SE#USE RE#NAME RE#REQ RE#CUR) NOWRITE"

    Do i = 1 to record.Ø
     Parse Var record.i sys#name se#name se#avl se#use re#name re#req re#cur .
     If sys#name = sys#prev Then Do
        sys#name = ''
        If se#name = se#prev Then Do
           se#name = ''
           se#avl  = ''
           se#use  = ''
```

```
        End
        Else se#prev = se#name
     End
     Else Do
        sys#prev = sys#name
        se#prev  = se#name
     End

    "ISPEXEC TBADD WLMSERE"
  End

 "ISPEXEC TBTOP WLMSERE"
 "ISPEXEC TBSKIP WLMSERE NUMBER("offset")"
 "ISPEXEC TBDISPL WLMSERE PANEL(WLMSERE)"
  retcode = RC
  offset  = ztdtop
 "ISPEXEC TBEND WLMSERE"
End
Exit
```

PANEL

```
)ATTR
 \ TYPE(TEXT) INTENS(LOW) HILITE(USCORE)
 ¬ TYPE(OUTPUT) INTENS(HIGH) CAPS(ON)
 @ TYPE(OUTPUT) INTENS(LOW) CAPS(ON)
 } TYPE(OUTPUT) INTENS(LOW) CAPS(ON) JUST(RIGHT)
)BODY EXPAND([])
+[-]\WLM Scheduling Environments & Resources+[-]
%COMMAND ===>_ZCMD
%SCROLL===>_ZUSC+
+
+
%System  ===>_SYSTEM  +         (System Name, Blank or '*')
%SCHENV  ===>_SCHENV          + (Scheduling Environment, Blank or '*')
+
+
\System  \SCHENV Name     \Avl \Users+\Resource Name   \Reqstat
\Curstat+
)MODEL
¬SYS#NAME¬SE#NAME          @Z  }Z   + ¬RE#NAME         @RE#REQ  @RE#CUR
)INIT
.ZVARS = '(SE#AVL SE#USE)'
)PROC
)END
```

Alain Callebaut
Systems Programmer
National Bank of Belgium

Generating new character sets

INTRODUCTION

We have a lot of publications that need to be printed at our installation. We use standard hardware and software platforms for host publishing such as 3820 printers, PSF, DCF, GML, GDDM, and FLSF. We chose bitmap fonts from the standard SYS1.FONTLIBB library at the beginning, but our users sometimes require character sets that do not exist in the official IBM character sets. That is why we developed a procedure for increasing and decreasing character set size, starting from the existing character set. By using this procedure we have made many new character sets which range in size from 5 to 70 points. The quality of these character sets is equivalent to source character sets.

Jobs FLSFZOO1, FLSFZOO2, and FLSFZOO3 are generated by the job FLSFZGEN, which produces the new character sets. The core of the process, program FONTZOOM, makes the specified size change and more. A serious problem in the process of increasing character size is the deformation of curves. The program solves that problem by redesigning each target character, and making the curves round.

There are some national characters in our alphabet that are present in some types of character sets, but many of the interesting character sets do not contain them. That is why we use the program FONTZOOM to increase or decrease specified national characters and add them in character set without them. The job FLSFZCH makes characters of the required size and adds them to a character set.

JOB FLSFZGEN

FLSFZGEN generates three jobs and the result of their submission is the new character set. Please take note of the comments provided, they will provide considerable help for users.

```
//useridG JOB NOTIFY=&SYSUID,
//           CLASS=A,MSGCLASS=X,MSGLEVEL=(2,Ø)
//*****************************************************************
//** JOB GENERATES THREE JOBS FOR DEFINING TARGET INCREASED OR      ***
//** DECREASED CHARACTER SET STARTING FROM SOURCE CHARACTER SET     ***
```

```
//**  1) FLSFZ001 - EXTRACTS CHARACTERS INTO PATTERNS               ***
//**  2) FLSFZ002 - INCREASES OR DECREASES CHARACTERS               ***
//**                ACCORDING TO INPUT PARAMETERS                   ***
//**  3) FLSFZ003 - DEFINES NEW CHARACTER SET UNDER SPECIFIED NAME  ***
//**                                                                ***
//** - YOU HAVE TO MAKE FOLLOWING CHANGES:                         ***
//**   C 'SYS1.FONTLIBB' 'FONT LIBRARY' ALL                         ***
//**   C 'CØSØLR12' 'SOURCE CHARACTER SET NAME' ALL                 ***
//**   C 'CØSØLR15' 'TARGET CHARACTER SET NAME' ALL                 ***
//** - SPECIFY PARAMETERS FOR ZOOMING IN GENJOB2 STEP              ***
//**   FOR EXAMPLE: 1Ø,8 MEANS THAT 1Ø POINTS FONT IS DECREASED TO  ***
//**   8 POINTS                                                     ***
//** - PAY ATTENTION TO DEFNEWF STEP IN FLSFZ003 JOB, WHERE        ***
//**   CHARACTERISTICS OF THAT CHARACTER SET IS SPECIFIED            ***
//**                                                                ***
//**      RECOMENDATION: BEST RESULTS ARE ACHIEVED WHEN DECREASES   ***
//**      AND INCREASES ARE NOT TO LARGE                            ***
//**                                                                ***
//********************************************************************
//LISTFONT    EXEC  PGM=AFLSF,PARM='PRINTER=382Ø,CONTINUE=YES'
//STEPLIB     DD DSN=FLSF.FN1LOAD,DISP=SHR
//            DD DSN=FLSF.AFN1LOAD,DISP=SHR
//SYSPRINT    DD SYSOUT=X
//FONTLIB     DD DSN=SYS1.FONTLIBB,DISP=SHR
//SYSOUT      DD DSN=&&FONTS,DISP=(NEW,PASS),
//            UNIT=SYSDA,DCB=(RECFM=FB,LRECL=133,BLKSIZE=798Ø),
//            SPACE=(TRK,(1Ø,5),RLSE)
//SYSIN       DD *
  LIST CH=* FONT=CØSØLR12
/*
//SORTN   EXEC PGM=SORT,REGION=ØK
//SORTIN   DD DSN=&&FONTS,DISP=(SHR,PASS)
//SORTOUT   DD DSN=&&FONTSPRM,DISP=(NEW,PASS),
//          UNIT=SYSDA,DCB=(RECFM=FB,LRECL=8Ø,BLKSIZE=312Ø),
//           SPACE=(TRK,(1Ø,5),RLSE)
//SORTOUT  DD SYSOUT=X
//SORTWKØ1 DD UNIT=SYSDA,SPACE=(CYL,1)
//SORTWKØ2 DD UNIT=SYSDA,SPACE=(CYL,1)
//SYSPRINT DD SYSOUT=X
//SYSOUT   DD SYSOUT=X
//SYSIN DD *
 INCLUDE COND=(2,1,NE,C' ',AND,1Ø,4,EQ,C'    '),FORMAT=CH
 SORT FIELDS=(1,9,A),FORMAT=CH
 SUM  FIELDS=NONE
 END
/*
//GENFLSFP EXEC PGM=ICETOOL,REGION=1M
//TOOLMSG  DD SYSOUT=X
//DFSMSG   DD SYSOUT=X
//IN        DD DSN=&&FONTSPRM,DISP=SHR
//OUT1      DD DSN=&&EXTZOOM,DISP=(NEW,PASS),
//          UNIT=SYSDA,DCB=(RECFM=FB,LRECL=8Ø,BLKSIZE=312Ø),
```

```
//          SPACE=(TRK,(1,1))
//OUT2      DD DSN=&&FLSFJ,DISP=(MOD,PASS),
//          UNIT=SYSDA,DCB=(RECFM=FB,LRECL=8Ø,BLKSIZE=312Ø),
//           SPACE=(TRK,(1Ø,5),RLSE)
//OUT3      DD DSN=&&DEFINE,DISP=(NEW,PASS),
//           UNIT=SYSDA,DCB=(RECFM=FB,LRECL=8Ø,BLKSIZE=312Ø),
//           SPACE=(TRK,(1Ø,5),RLSE)
//TOOLIN   DD *
  COPY FROM(IN) TO(OUT1) USING(EXTF)
  COPY FROM(IN) TO(OUT2) USING(ZOM1)
  COPY FROM(IN) TO(OUT2) USING(ZOM2)
  COPY FROM(IN) TO(OUT3) USING(DEFF)
/*
//*** EXTRACTING FONTS
//EXTFCNTL DD *
  OUTREC FIELDS=(C' EXTRACT FONT=CØSØLR12 CH=',2,8,C':',2,8,4ØX)
/*
//ZOM1CNTL DD *
  OUTREC FIELDS=(C'//',2,5,C'I',8,2,
   C'  DD DSN=userid.FLSF.PATTERN.CØSØLR12(',2,8,C'),DISP=SHR',4ØX)
/*
//ZOM2CNTL DD *
  OUTREC FIELDS=(C'//',2,5,C'O',8,2,
   C'  DD DSN=userid.FLSF.PATTERN.CØSØLR15(',2,8,C'),DISP=SHR',4ØX)
/*
//*** DEFINE NEW FONTS
//DEFFCNTL DD *
  OUTREC FIELDS=(C' DEFINE FONT=CØSØLR15 CH=',2,8,C':',2,8,4ØX)
/*
//GENJOB1   EXEC PGM=IDCAMS
//SYSPRINT  DD SYSOUT=X
//SYSUT1    DD DATA,DLM=XX  **********
//userid1     JOB MSGLEVEL=(Ø,Ø),MSGCLASS=X,NOTIFY=&SYSUID
//*********************************************************************
//**  DELETE PATTERN DATASET FOR OLD FONT                             **
//*********************************************************************
//IDCAMS   EXEC PGM=IDCAMS,REGION=1M
//SYSPRINT DD   SYSOUT=*
//SYSIN    DD   *
  DELETE   userid.FLSF.PATTERN.CØSØLR12
  SET MAXCC=Ø
/*
//*********************************************************************
//**  EXTRACT OLD FONT INTO PATERN DATASET                            **
//*********************************************************************
//EXTRATF     EXEC PGM=AFLSF,PARM='PRINTER=382Ø,CONTINUE=YES'
//STEPLIB     DD DSN=FLSF.FN1LOAD,DISP=SHR
//            DD DSN=FLSF.AFN1LOAD,DISP=SHR
//SYSPRINT    DD SYSOUT=X
//FONTLIB     DD DSN=SYS1.FONTLIBB,DISP=SHR
//PATTLIB     DD DSN=userid.FLSF.PATTERN.CØSØLR12,DISP=(NEW,CATLG),
//            UNIT=SYSDA,DCB=(RECFM=VB,LRECL=133,BLKSIZE=82Ø9),
```

```
//            SPACE=(CYL,(5,5,5Ø),RLSE)
//SYSOUT      DD SYSOUT=X
//SYSIN       DD *
XX
//            DD DSN=&&EXTZOOM,DISP=SHR
//SYSUT2      DD DSN=userid.USER.CNTL(FLSFZOO1),DISP=SHR
//SYSIN       DD *
  REPRO IFILE(SYSUT1) OFILE(SYSUT2)
/*
//*********************************************************************
//**  PART FOR ZOOMING FONTS
//*********************************************************************
//GENJOB2   EXEC PGM=IDCAMS
//SYSPRINT  DD SYSOUT=X
//SYSUT1    DD DATA,DLM=XX  **********
//userid2 JOB (ACCT#),'D.N',
//           NOTIFY=&SYSUID,
//           CLASS=C,MSGCLASS=X,MSGLEVEL=(Ø,Ø)
//*********************************************************************
//**  DELETE PATTERN DATASET FOR TARGET FONT                         **
//*********************************************************************
//IDCAMS   EXEC PGM=IDCAMS,REGION=1M
//SYSPRINT DD   SYSOUT=*
//SYSIN    DD   *
  DELETE   userid.FLSF.PATTERN.CØSØLR15
  SET MAXCC=Ø
/*
//*********************************************************************
//**  ALLOC PATTERN DATASET FOR TARGET FONT                          **
//*********************************************************************
//ALLOC   EXEC PGM=IEFBR14
//PATTLIB     DD DSN=userid.FLSF.PATTERN.CØSØLR15,DISP=(NEW,CATLG),
//            UNIT=SYSDA,DCB=(RECFM=VB,LRECL=133,BLKSIZE=82Ø9),
//            SPACE=(CYL,(5,5,5Ø),RLSE)
//*********************************************************************
//**  PROGRAM INCREASES OR DECREASES CHARACTERS ACCORDING TO INPUT  **
//**  PARAMETERS THAT MEANS SIZE OF SOURCE CHARACTERS, SIZE OF      **
//**  TARGET CHARACTERS                                             **
//*********************************************************************
//FONTZOOM EXEC PGM=FONTZOOM,PARM='/Ø9,Ø8'
//STEPLIB  DD DSN=userid.USER.LOAD,DISP=SHR
//SYSPRINT DD SYSOUT=X
XX
//            DD DSN=&&FLSFJ,DISP=SHR
//            DD DATA,DLM=YY  **********
//SYSIN    DD *
YY
//            DD DSN=&&FONTSPRM,DISP=SHR
//SYSUT2      DD DSN=userid.USER.CNTL(FLSFZOO2),DISP=SHR
//SYSIN       DD *
  REPRO IFILE(SYSUT1) OFILE(SYSUT2)
/*
```

```
//*********************************************************************
//**  PART THAT GENERATES JOB FOR DEFINING NEW MAPS IN FONTS         **
//*********************************************************************
//GENJOB3   EXEC PGM=IDCAMS
//SYSPRINT  DD SYSOUT=X
//SYSUT1    DD DATA,DLM=XX  **********
//userid3     JOB MSGLEVEL=(Ø,Ø),MSGCLASS=X,NOTIFY=&SYSUID
//*********************************************************************
//**  DEFINE NEW CHARACTER SET FROM PATERN DATASET IN SYS1.FONTLIBB **
//**  NOTE:                                                          **
//**  DEF FONT DEFINES GENERAL CHARACTERISTICS OF SPECIFIED          **
//**  CHARACTER SET. CHECK PARAMETER VALUES BEFORE SUBMITING JOB.    **
//**  THIS IS A SAMPLE DEFINITION FOR FONT FAMILY LETTER GOTHIC,     **
//**  POINTSIZE 8.Ø, MEDIUM NORMAL, NO MONOSPACED                    **
//*********************************************************************
//DEFNEWF     EXEC PGM=AFLSF,PARM='PRINTER=382Ø,CONTINUE=YES'
//STEPLIB     DD DSN=FLSF.FN1LOAD,DISP=SHR
//             DD DSN=FLSF.AFN1LOAD,DISP=SHR
//SYSPRINT    DD SYSOUT=X
//FONTLIB     DD DSN=SYS1.FONTLIBB,DISP=SHR
//PATTLIB     DD DSN=userid.FLSF.PATTERN.CØSØLR15,DISP=SHR
//SYSOUT      DD SYSOUT=X
//SYSIN       DD *
 /* UNCOMMENT NEXT LINE IN CASE OF REPETITION */
 /* DELETE FONT=CØSØLR15  */
 DEFINE FONT=CØSØLR15 -
        FAMILY='LETTER GOTHIC' -
        POINTSIZE=8.Ø     -
        SLOPE=Ø:Ø         -
        WEIGHT=MEDIUM     -
        WIDTH=NORMAL      -
        ITALIC=NO         -
        LINESPACE=3Ø      -
        WORDSPACE=16      -
        FIGSPACE=16       -
        MHEIGHT=18        -
        MSPACE=16         -
        NSPACE=Ø          -
        EHEIGHT=Ø         -
        XHEIGHT=13        -
        UPOSITION=2       -
        UWIDTH=2          -
        MONOSPACED=NO     -
        NEGATIVE=NO       -
        OUTLINE=NO        -
        OVERSTRUCK=NO     -
        UNDERSCORE=NO
XX
//             DD DSN=&&DEFINE,DISP=SHR
//SYSUT2      DD DSN=userid.USER.CNTL(FLSFZOO3),DISP=SHR
//SYSIN       DD *
  REPRO IFILE(SYSUT1) OFILE(SYSUT2)
/*
```

JOB FLSFZCH

FLSFZCH adds only specified characters to an existing character set by increasing or decreasing specified source characters.

```
//useridC     JOB MSGLEVEL=(1,1),MSGCLASS=X,NOTIFY=&SYSUID
//*********************************************************************
//** JOB INCREASES OR DECREASES SPECIFIED NATIONAL CHARACTERS AND   ***
//** ADD THEM TO CHARACTER SET THAT DOESN'T CONTAIN THEM INITIALLY  ***
//**                                                               ***
//** - YOU HAVE TO MAKE FOLLOWING CHANGES:                         ***
//**   C 'SYS1.FONTLIBB' 'FONT LIBRARY' ALL                       ***
//**   C 'CØSØLR12' 'SOURCE CHARACTER SET NAME' ALL               ***
//**   C 'CØSØLR15' 'TARGET CHARACTER SET NAME' ALL               ***
//** - SPECIFY PARAMETERS FOR ZOOMING IN GENJOB2 STEP              ***
//**   FOR EXAMPLE: 1Ø,8 MEANS THAT 1Ø POINTS FONT IS DECREASED TO  ***
//**   8 POINTS                                                    ***
//**                                                               ***
//** - DEFINE FONTS THAT YOU PROCESS AND ADD THEM IN SOME CHARACTER ***
//**   SET. FOR EXAMPLE:                                          ***
//**        LE12ØØØØ - CAPITAL LETTERS                             ***
//**        LE11ØØØØ - SMALL LETTERS                               ***
//*********************************************************************
//*********************************************************************
//**  DELETE PATTERN DATASET FOR SOURCE FONT                        **
//*********************************************************************
//IDCAMS   EXEC PGM=IDCAMS,REGION=1M
//SYSPRINT DD   SYSOUT=*
//SYSIN    DD   *
  DELETE   userid.FLSF.PATTERN.CØSØCR12
  DELETE   userid.FLSF.PATTERN.CØSØCR15
  SET MAXCC=Ø
/*
//*********************************************************************
//**  EXTRACT SOURCE FONT INTO PATERN DATASET                       **
//*********************************************************************
//EXTRATF     EXEC PGM=AFLSF,PARM='PRINTER=382Ø,CONTINUE=YES'
//STEPLIB     DD DSN=FLSF.FN1LOAD,DISP=SHR
//            DD DSN=FLSF.AFN1LOAD,DISP=SHR
//SYSPRINT    DD SYSOUT=X
//FONTLIB     DD DSN=SYS1.FONTLIBB,DISP=SHR
//PATTLIB     DD DSN=userid.FLSF.PATTERN.CØSØCR12,DISP=(NEW,CATLG),
//            UNIT=SYSDA,DCB=(RECFM=VB,LRECL=133,BLKSIZE=82Ø9),
//            SPACE=(CYL,(5,5,5Ø),RLSE)
//SYSOUT      DD SYSOUT=X
//SYSIN       DD *
 EXTRACT FONT=CØSØCR12 CH=LE12ØØØØ:LE12ØØØØ
 EXTRACT FONT=CØSØCR12 CH=LE11ØØØØ:LE11ØØØØ
/*
//LISTFONT    EXEC  PGM=AFLSF,PARM='PRINTER=382Ø,CONTINUE=YES'
//STEPLIB     DD DSN=FLSF.FN1LOAD,DISP=SHR
//            DD DSN=FLSF.AFN1LOAD,DISP=SHR
```

```
//SYSPRINT    DD SYSOUT=X
//FONTLIB     DD DSN=SYS1.FONTLIBB,DISP=SHR
//SYSOUT      DD DSN=&&FONTS,DISP=(NEW,PASS),
//            UNIT=SYSDA,DCB=(RECFM=FB,LRECL=133,BLKSIZE=7980),
//            SPACE=(TRK,(10,5),RLSE)
//SYSIN       DD *
  LIST FONT=C0S0CR12 CH=LE120000
  LIST FONT=C0S0CR12 CH=LE110000
/*
//SORTN    EXEC PGM=SORT,REGION=0K
//SORTIN   DD DSN=&&FONTS,DISP=(SHR,PASS)
//SORTOUT  DD DSN=&&FONTSPRM,DISP=(NEW,PASS),
//          UNIT=SYSDA,DCB=(RECFM=FB,LRECL=80,BLKSIZE=3120),
//          SPACE=(TRK,(10,5),RLSE)
//SORTOUT  DD SYSOUT=X
//SYSPRINT DD SYSOUT=X
//SYSOUT   DD SYSOUT=X
//SYSIN DD *
 INCLUDE COND=(2,1,NE,C' ',AND,10,4,EQ,C'    '),FORMAT=CH
 SORT FIELDS=(1,9,A),FORMAT=CH
 SUM  FIELDS=NONE
 END
/*
//*********************************************************************
//**  ALLOC PATTERN DATASET FOR NEW FONT                             **
//*********************************************************************
//ALLOC   EXEC PGM=IEFBR14
//PATTLIB     DD DSN=userid.FLSF.PATTERN.C0S0CR15,DISP=(NEW,CATLG),
//            UNIT=SYSDA,DCB=(RECFM=VB,LRECL=133,BLKSIZE=8209),
//            SPACE=(CYL,(5,5,50),RLSE)
//*********************************************************************
//**  PROGRAM FOR DECREASING OR INCREASING CHARACTERS ACCORDING     **
//**  TO PARAMETERS. IN THIS EXAMPLE, CHARACTERS DECREASED FROM     **
//**  9 TO 8 POINTS                                                 **
//*********************************************************************
//FONTZOOM  EXEC PGM=FONTZOOM,PARM='/09,08'
//STEPLIB   DD DSN=userid.USER.LOAD,DISP=SHR
//SYSPRINT  DD SYSOUT=X
//LE120I00  DD DSN=userid.FLSF.PATTERN.C0S0CR12(LE120000),DISP=SHR
//LE120O00  DD DSN=userid.FLSF.PATTERN.C0S0CR15(LE120000),DISP=SHR
//LE110I00  DD DSN=userid.FLSF.PATTERN.C0S0CR12(LE110000),DISP=SHR
//LE110O00  DD DSN=userid.FLSF.PATTERN.C0S0CR15(LE110000),DISP=SHR
//SYSIN     DD DSN=&&FONTSPRM,DISP=SHR
/*
//*********************************************************************
//**  DEFINE NEW FONT FROM PATERN DATASET INTO SYS1.FONTLIBB         **
//*********************************************************************
//DEFNEWF     EXEC PGM=AFLSF,PARM='PRINTER=3820,CONTINUE=YES'
//STEPLIB     DD DSN=FLSF.FN1LOAD,DISP=SHR
//            DD DSN=FLSF.AFN1LOAD,DISP=SHR
//SYSPRINT    DD SYSOUT=X
//FONTLIB     DD DSN=SYS1.FONTLIBB,DISP=SHR
```

```
//PATTLIB     DD DSN=userid.FLSF.PATTERN.CØSØCR15,DISP=SHR
//SYSOUT      DD SYSOUT=X
//SYSIN       DD *
 DEFINE FONT=CØSØCR15  CH=LE120000:LE120000
 DEFINE FONT=CØSØCR15  CH=LE110000:LE110000
 /* DEFINITION OF CHARACTERS IN CODE PAGE           */
 /* DELETE CPAGE=T1SØCR15 CPOINT=49                 */
 /* DELETE CPAGE=T1SØCR15 CPOINT=47                 */
 /* DEFINE CPAGE=T1SØCR15 CPOINT=49 CH=LE120000     */
 /* DEFINE CPAGE=T1SØCR15 CPOINT=47 CH=LE110000     */
/*
//
```

FONTZOOM

FONTZOOM has two input parameters passed by the PARM option. The first parameter specifies the size of the source characters, the second targets character size. Another input for this program is in the SYSIN dataset; you can see it in the FLSFZOO2 generated job. This data represents the size that characterizes each character. They are output from the AFLSF program that is executed at the beginning of job FLSFGEN. Using these inputs, FONTZOOM makes the specified size change and any redesign required. Output of FONTZOOM is the image of the target characters and all parameters necessary for defining each individual character.

```
FONTZOOM: PROCEDURE(PARM) OPTIONS(MAIN) REORDER;
/*****************************************************************/
/* PROGRAM DECRESES OR INCREASES CHARACTERS IN PATTLIB            */
/* PARAMETERS ARE                                                */
/*         P1 - SOURCE CHARACTER SIZE                            */
/*         P2 - TARGET CHARACTER SIZE                            */
/*****************************************************************/
DCL IN       FILE SEQL    INPUT;
DCL OUT      FILE SEQL    OUTPUT;
DCL PARM  CHAR(1Ø) VAR;
DCL RECORD CHAR(133) VAR;
DCL AREA CHAR(133) BASED(P), P PTR;
DCL MATS(YS,XS)  CHAR(1) CTL;
DCL MATN(YN,XN)  CHAR(1) CTL;
DCL MATZ(YN,XN)  BIN FIXED CTL;
DCL MATT(YN,XN)  BIN FIXED CTL;
DCL RECORDN      CHAR(XN) CTL;
DCL CHAR1 CHAR(1);
DCL CHAR1S CHAR(1);
DCL COMPASS CHAR(2);
DCL J BIN FIXED;
DCL (XS,YS) BIN FIXED;
DCL (XN,YN) BIN FIXED;
```

```
DCL NAME CHAR(8);
DCL (ASP,CSP) BIN FIXED;
DCL (TOP,BOT) BIN FIXED;
DCL OFF BIN FIXED;
DCL BSP BIN FIXED;
DCL HEIGHT BIN FIXED;
DCL MAP CHAR(8);
DCL NEOF  BIT;
DCL NEOFS INIT('1'B) BIT;
DCL (MIN,MAX,SUBSTR,INDEX,TRUNC,ABS) BUILTIN;
ON ENDFILE(IN) NEOF='Ø'B;
ON ENDFILE(SYSIN) NEOFS='Ø'B;
ON ERROR SNAP SYSTEM;

GET STRING(PARM) LIST(P1,P2);
Q=P2/P1;
GET SKIP EDIT(MAP) (X(1),A(8)) COPY;
DO WHILE(NEOFS);
OPEN FILE(IN) TITLE(SUBSTR(MAP,1,5)||'I'||SUBSTR(MAP,7,2)),
     FILE(OUT) TITLE(SUBSTR(MAP,1,5)||'O'||SUBSTR(MAP,7,2));
NEOF='1'B;
J=Ø;
READ FILE(IN) INTO(RECORD);
NAME=SUBSTR(RECORD,INDEX(RECORD,'NAME')+5,8);
GET STRING(SUBSTR(RECORD,INDEX(RECORD,'ASP')+5,4)) LIST(ASP);
GET STRING(SUBSTR(RECORD,INDEX(RECORD,'CSP')+5,4)) LIST(CSP);
GET STRING(SUBSTR(RECORD,INDEX(RECORD,'TOP')+5,4)) LIST(TOP);
GET STRING(SUBSTR(RECORD,INDEX(RECORD,'BOT')+5,4)) LIST(BOT);
GET STRING(SUBSTR(RECORD,INDEX(RECORD,'OFF')+5,4)) LIST(OFF);
GET STRING(SUBSTR(RECORD,INDEX(RECORD,'BSP')+5,4)) LIST(BSP);
GET STRING(SUBSTR(RECORD,INDEX(RECORD,'HEIGHT')+8,4)) LIST(HEIGHT);
CHAR1S='.';
YS=HEIGHT;
XS=BSP;
K=(ASP+XS+CSP)*Q+Ø.5;
L=(TOP+YS+BOT)*Q+Ø.5;
XN=XS*Q+Ø.5;
ASP=ASP*Q;
CSP=K-XN-ASP;
IF CSP< Ø & ASP > Ø
THEN DO;
     ASP=ASP-CSP;
     CSP=Ø;
     END;
YN=YS*Q+Ø.5;
TOP=TOP*Q;
BOT=L-YN-TOP;
IF BOT< Ø & TOP > Ø
THEN DO;
     TOP=TOP-BOT;
     BOT=Ø;
     END;
```



```
OFF=OFF*Q+Ø.5;
IF OFF = Ø
THEN OFF = 1;
ALLOC MATS;
PUT STRING(SUBSTR(RECORD,INDEX(RECORD,'ASP')+4,3)) EDIT(ASP) (P'S99');
PUT STRING(SUBSTR(RECORD,INDEX(RECORD,'CSP')+4,3)) EDIT(CSP) (P'S99');
PUT STRING(SUBSTR(RECORD,INDEX(RECORD,'TOP')+4,3)) EDIT(TOP) (P'S99');
PUT STRING(SUBSTR(RECORD,INDEX(RECORD,'BOT')+4,3)) EDIT(BOT) (P'S99');
PUT STRING(SUBSTR(RECORD,INDEX(RECORD,'OFF')+4,4)) EDIT(OFF) (P'S999');
PUT STRING(SUBSTR(RECORD,INDEX(RECORD,'BSP')+4,4)) EDIT(XN) (P'S999');
PUT STRING(SUBSTR(RECORD,INDEX(RECORD,'HEIGHT')+7,4)) EDIT(YN)
(P'S999');
WRITE FILE(OUT) FROM(RECORD);
ALLOC MATN,RECORDN;
MATN(*,*)='.';
READ FILE(IN) INTO(RECORD);
DO I=1 BY 1 WHILE(NEOF);
   P=ADDR(MATS(I,1));
   SUBSTR(AREA,1,BSP)=RECORD;
   READ FILE(IN) INTO(RECORD);
END;
ALLOC MATZ,MATT;
MATZ(*,*)=Ø;
MATT(*,*)=Ø;
IF Q < 1
THEN                  /* DECREASING */
DO;
     DO I=1 TO YS;
        K=MAX(1,I*Q+Ø.5);
        DO J=1 TO XS;
           L=MAX(1,J*Q+Ø.5);
           IF MATS(I,J) = '*'
           THEN MATZ(K,L) = MATZ(K,L)+1;
           ELSE MATT(K,L) = MATT(K,L)+1;
        END;
     END;
     DO I=1 TO YN;
        DO J=1 TO XN;
           IF MATZ(I,J) > Ø & MATZ(I,J) >= MATT(I,J)
           THEN MATN(I,J) = '*';
        END;
     END;
END;
ELSE                  /* INCREASING */
DO I=1 TO YS;
   DO J=1 TO XS;
      CHAR1=MATS(I,J);
      COMPASS=' ';
      IF CHAR1 ¬= CHAR1S
      THEN
      IF I=1 & J > 1 & J < XS &
      MATS(I,J-1) ¬= CHAR1 & MATS(I,J+1) = CHAR1 &   /*  .?*  */
```

```
MATS(I+1,J) = CHAR1 & MATS(I+1,J+1) = CHAR1    /*  X**  */
THEN COMPASS='DD';
ELSE
IF I=1 & J > 1 & J < XS &
MATS(I,J-1) = CHAR1 & MATS(I,J+1) ¬= CHAR1 &   /*  *?.  */
MATS(I+1,J-1) = CHAR1 & MATS(I+1,J) = CHAR1    /*  **X  */
THEN COMPASS='DL';
ELSE
IF I=YS & J > 1 & J < XS &
MATS(I-1,J) = CHAR1 & MATS(I-1,J+1) = CHAR1 &   /*  X**  */
MATS(I,J-1) ¬= CHAR1 & MATS(I,J+1) = CHAR1      /*  .?*  */
THEN COMPASS='GD';
ELSE
IF I=YS & J > 1 & J < XS &
MATS(I-1,J-1) = CHAR1 & MATS(I-1,J) = CHAR1 &   /*  **X  */
MATS(I,J-1) = CHAR1 & MATS(I,J+1) ¬= CHAR1      /*  *?.  */
THEN COMPASS='GL';
ELSE
IF I > 1 & I < YS  & J = 1 &
MATS(I-1,J) = CHAR1 & MATS(I-1,J+1) = CHAR1 &   /*  **  */
MATS(I,J+1) = CHAR1 & MATS(I+1,J+1) ¬= CHAR1    /*  ?*  */
THEN COMPASS='GD';                              /*  .X  */
ELSE
IF I > 1 & I < YS  & J = 1 &
MATS(I-1,J) ¬= CHAR1 & MATS(I,J+1) = CHAR1 &    /*  .X  */
MATS(I+1,J) = CHAR1 & MATS(I+1,J+1) = CHAR1     /*  ?*  */
THEN COMPASS='DD';                              /*  **  */
ELSE
IF I > 1 & I < YS  & J = XS &
MATS(I,J-1) = CHAR1 & MATS(I-1,J) ¬= CHAR1 &    /*  X.  */
MATS(I+1,J-1) = CHAR1 & MATS(I+1,J) = CHAR1     /*  *?  */
THEN COMPASS='DL';                              /*  **  */
ELSE
IF I > 1 & I < YS  & J = XS &
MATS(I-1,J-1) = CHAR1 & MATS(I-1,J) = CHAR1 &   /*  **  */
MATS(I,J-1) = CHAR1 & MATS(I+1,J) ¬= CHAR1      /*  *?  */
THEN COMPASS='GL';                              /*  X.  */
ELSE
IF I > 1 & I < YS & J > 1 & J < XS &
MATS(I-1,J-1) ¬= CHAR1 & MATS(I-1,J) ¬= CHAR1 & /*   ..X  */
MATS(I,J-1) ¬= CHAR1  & MATS(I,J+1) = CHAR1 &   /*   .?*  */
MATS(I+1,J) = CHAR1   & MATS(I,J-1) = CHAR1     /*   X**  */
THEN COMPASS='DD';
ELSE
IF I > 1 & I < YS & J > 1 & J < XS &
MATS(I-1,J-1) = CHAR1 & MATS(I-1,J) = CHAR1 &   /*   **X  */
MATS(I,J-1) = CHAR1  & MATS(I,J+1) ¬= CHAR1 &   /*   *?.  */
MATS(I+1,J) ¬= CHAR1 & MATS(I,J-1) ¬= CHAR1     /*   X..  */
THEN COMPASS='GL';
ELSE
IF I > 1 & I < YS & J > 1 & J < XS &
MATS(I-1,J+1) ¬= CHAR1 & MATS(I-1,J) ¬= CHAR1 & /*   X..  */
```

```
      MATS(I,J-1) = CHAR1   & MATS(I,J+1) ¬= CHAR1 &  /*   *?.  */
      MATS(I+1,J+1) = CHAR1 & MATS(I+1,J) = CHAR1     /*   **X  */
      THEN COMPASS='DL';
      ELSE
      IF I > 1 & I < YS & J > 1 & J < XS &
      MATS(I-1,J+1) = CHAR1 & MATS(I-1,J) = CHAR1 &   /*   X**  */
      MATS(I,J-1) ¬= CHAR1   & MATS(I,J+1) = CHAR1 &  /*   .?*  */
      MATS(I+1,J+1) ¬= CHAR1 & MATS(I+1,J) ¬= CHAR1   /*   ..X  */
      THEN COMPASS='GD';
      IY=(I-1)*Q+Ø.5;
      JX=(J-1)*Q+Ø.5;
      SELECT(COMPASS);
      WHEN('DD')  DO I1=IY+1 TO I*Q+Ø.5;
                  DO J1=J*Q+Ø.5-(I1-IY)*Q/Q/2 TO J*Q+Ø.5;
                     MATN(I1,J1)=CHAR1;
                  END;
                  END;
      WHEN('DL')  DO I1=IY+1 TO I*Q+Ø.5;
                  DO J1=JX+1 TO JX+1+(I1-IY)*Q/Q/2;
                     MATN(I1,J1)=CHAR1;
                  END;
                  END;
      WHEN('GD')  DO I1=IY+1 TO I*Q+Ø.5;
                  DO J1=JX+1+(I1-IY)*Q/Q/2 TO J*Q+Ø.5;
                     MATN(I1,J1)=CHAR1;
                  END;
                  END;
      WHEN('GL')  DO I1=IY+1 TO I*Q+Ø.5;
                  DO J1=JX+1 TO J*Q+Ø.5-(I1-IY)*Q/Q/2;
                     MATN(I1,J1)=CHAR1;
                  END;
                  END;
      OTHERWISE   DO I1=IY+1 TO I*Q+Ø.5;
                  DO J1=JX+1 TO J*Q+Ø.5;
                     MATN(I1,J1)=CHAR1;
                  END;
                  END;
     END; /* SELECT */
   END;
END;
DO I=1 TO YN;
   RECORDN=STRING(MATN(I,*));
   WRITE FILE(OUT) FROM(RECORDN);
END;
FREE MATS,MATN,RECORDN,MATZ,MATT;
CLOSE FILE(IN), FILE(OUT);
GET SKIP EDIT(MAP) (X(1),A(8)) COPY;
END;
END FONTZOOM;
```

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

z/OS: future directions

INTRODUCTION

In the April edition of *MVS Update* we reviewed the announcement of Version 1 Release 2 of z/OS, which is due for general release in October 2001 (see announcement letter ZP01-0164). In this article we consider how the functionality found in forthcoming releases of z/OS will have a profound effect on long-term, end-user strategies.

HARDWARE REQUIREMENTS

In the short term there will not be a new Architectural Level Set for z/OS Release 3. As with Release 1 and 2 of z/OS, Version 1 Release 3 will run on the z900 or comparable server, Generation 5 (G5) and Generation 6 (G6) S/390 Parallel Enterprise Servers, and all models of the Multiprise 3000 Enterprise Server. For a complete overview of z/OS Version 1 Release 2 software prerequisites, refer to the *z/OS Planning for Installation* (GA22-7504) publication. Further information can be found at the following URL: http://www.ibm.com/servers/eserver/zseries/.

WORKLOAD LICENSE CHARGES (WLC)

A key part of the IBM announcement is the availability of Workload License Charges (usage-based pricing) with z/OS and the z900. This is a crucial part of the zServer package. The current pricing models penalize those with unused capacity, or those wishing to migrate large non-System/390 applications to the mainframe.

In general the new mainframe pricing structures represent a considerable improvement on previous models. In particular they allow for new applications to be implemented far more economically than before, reduce the incremental cost of existing applications and minimize the cost of running some typical legacy applications such as IMS. Furthermore, for the smaller users of around 200 to 500 MIPS, they greatly improve the current situation where the current level of processor granularity results in steep software cost increases when more capacity is needed.

However, there is a note of caution. The new pricing model will require a new outlook for users and operators:

- Users will need to keep track of how close they run to the licence limit. One of the benefits of z/OS is the ability to run at near 100% capacity. IBM allows for peaks that exceed the licence limit, but these peaks must not exceed four hours in duration, otherwise there will be a cost penalty. Monitoring the licence limit will be crucial.
- There is an operator command to change the licence level, and it is *absolutely crucial* that this command is RACF protected and there is a policy in place that indicates exactly who has responsibility for changes to this level. Because of the serious financial implications of changing the level, operators may wish to shift responsibility to senior management.

Therefore, the WLC and the extra capacity provided in the z900 boxes through CUoD will have a profound impact on the role of capacity planners. In the past capacity planners have been concerned with maintaining a balance between having too little and too much capacity. Usually this is achieved by installing more capacity than required, to insure against future performance problems.

Now with boxes containing more capacity than required and software licensing fees based on capacity defined, rather than physical resources installed, enterprises will be able to maintain considerable on-site capacity reserves. As a result, many of the traditional capacity planning problems will become less critical. However, 'usage-based pricing' does not eliminate the need for planning, but it does transform the penalty for poor planning from a performance problem to a financial problem.

PARALLEL SYSPLEX

There are signs that Parallel Sysplex itself is changing. Clues to the metamorphosis are appearing in z/OS Version 1 Release 2, where we see duplexing of the Coupling Facility (CF) structures, showing that the medium-term plan is to have all the coupling done in ICFs inside the machine. In the future we are likely to see large SNPs with multiple logical partitions.

REAL STORAGE SUPPORT

64-bit addressing provides considerable benefits for z/OS. These can be divided into improvements to real memory, integer arithmetic, and virtual memory. The implementation of the 64-bit z/Architecture eliminates any bottlenecks associated with a lack of addressable memory by making the addressing capability virtually unlimited (16 exabytes compared with the current capability of 2 gigabytes). Both DB2 Version 6 (with PTF) and IMS Version 7 are enhanced to exploit 64-bit real storage above 2GB. Additionally, access methods such as BSAM, QSAM, and VSAM, Hierarchical File System (HFS), and Extended Remote Copy (XRC) have been enhanced to exploit 64-bit real storage above 2GB.

From z/OS Version 1 Release 3 there will be 64-bit virtual addressing for those applications that need it. This has implications for storage management tasks. With so much storage available users will be able to have as much as they require, so concepts such as subpools and 'above and below the line' will lose relevance.

PRIORITIZATION AND THE IRD

The use of the Intelligent Resource Director (IRD) with z/OS provides another indication of future directions. Already the IRD has three functions:

- *LPAR CPU management* – directing processor resources to priority workloads.
- *Dynamic Channel Path management* – delivering bandwidth to priority workloads.
- *Channel subsystem priority queueing* – where I/O queueing is directed by user goals.

In the mid-term future we are likely to see IRD technology used to dynamically expand and shrink expanded storage (memory) according to user priorities. Because this is technically challenging it should be expected within a two-year timeframe.

PERFORMANCE TUNING AND MODELLNG

z/OS comes with many new processing resources that will reduce the importance of queueing delays and congestion effects, but will not reduce the importance of analysing path lengths and processing times, or performing a root cause analysis, which is good news for performance analysts. Furthermore, the new mechanisms such as the Intelligent Resource Director will still have to be monitored carefully. In addition, the grouping of LPARs into clusters, and the initial configuration of logical processors, processing weights, and I/O priorities to LPARs will have to be done with care so that mission-critical workloads achieve desired performance levels.

IP VERSION 6

Support for IP Version 6 protocols will be a feature of future releases of z/OS. There are compelling reasons to replace the current Internet Protocol (IP Version 4) with the new generation protocol, called IP Version 6. The current protocol supports only a few billion different IP addresses, of which one billion can effectively be used under optimal circumstances. At the current rate of use the Internet protocol will run out of address spaces around 2004. This will be a prime mover towards a new protocol. IP Version 6 supports the almost infinite number of $8x10^{37}$ IP addresses, quite sufficient to provide every possible device with an IP address – even if the address space is used ineffectively. In fact we may get to a situation where IP addresses are seen as a disposable item.

In addition to solving the imminent IP addressing problem, IP Version 6 provides much better support for new network capabilities like broadband networking, IPSec, RSVP, and mobile IP. In spite of the obvious technological advantages, acceptance of IP Version 6 has been very slow. Today, there are only a few hundred sites that support the new protocol, versus more than a hundred million IP Version 4 Internet sites. It is, however, expected that interest in the new generation of IP will show massive growth in the coming years.

MVS news

Computer Associates has announced Version 10 of its CA-Datacom database for z/OS and OS/390, which enables data sharing throughout Parallel Sysplex.

Applications can access database information anywhere on any CA-Datacom image within the sysplex without any changes to the application code, providing fault-tolerance and cross-platform workload balancing. Version 10 has improved buffer and task management, enabing applications to exploit the virtual storage in z/OS and OS/390 and support, it is claimed, thousands of concurrent requests for data.

System availability improvements enable users to adjust system resources on the fly to address significant variations in system workloads without interrupting service. There are also improved database back-up and restore capabilities, expedited recovery, and streamlined message processing.

For further information contact:

Computer Associates International, Inc, One Computer Associates Plaza, Islandia, NY 11749, USA.
Tel: (631) 342 6000
Fax: (631) 342 6800

Computer Associates plc, Ditton Park, Riding Court Road, Datchet, Slough, Berkshire, SL3 9LL, UK.
Tel: 01753 577733
Fax: 01753 825464

http://www.ca.com

* * *

Most mainframe users foresee a healthy future for the IBM S/390 zSeries platform, according to Xephon's recent research. And, while some large customers are still not certain whether the mainframe is a viable platform for strategic new applications, others are relying on it to consolidate and manage their e-business systems.

We surveyed 59 managers at S/390 installations across the world, and found them generally very positive about the platform. 33% of respondents predicted strong long-term growth for the S/390 within their company, and a further 46% expected at least some growth for mainframe applications. Some users are still unsure about the cost benefits of moving to z/OS. 55% of our customers said they are already moving, or planning to move, to z/OS, but only 38% cited cost as the main benefit. Others were tempted by enhanced open systems compatibility, support for 64-bit apps, or workload management functions, and 31% saw it as a strategic direction.

When it came to usage-based pricing, nearly half of our sites were unsure whether the new cost structures in z/OS would save them money. Of those who have come to a conclusion, though, over two thirds believed that there would be cost savings in due course.

The full research analysis is published in Xephon's new report, *OS/390, z/OS, and the Future*. Further details can be found on our Web site at:

http:// www.xephon.com

* * *