136 VM

December 1997

update

In this issue

VM Update

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

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

Australian office
Xephon/RSM
GPO Box 6258
Halifax Street
Adelaide, SA 5000
Australia
Telephone: 088 223 1391

Contributions
Articles published in *VM Update* are paid for at the rate of £170 ($250) per 1000 words for original material. To find out more about contributing an article, without any obligation, please contact us at any of the addresses above and we will send you a copy of our *Notes for Contributors*.

Editor
Trevor Eddolls

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

Subscriptions and back-issues
A year's subscription to *VM Update*, comprising twelve monthly issues, costs £170.00 in the UK; $255.00 in the USA and Canada; £176.00 in Europe; £182.00 in Australasia and Japan; and £180.50 elsewhere. In all cases the price includes postage. Individual issues, starting with the January 1990 issue, are available separately to subscribers for £14.50 ($21.50) each including postage.

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

Printed in England.

EBCDIC to ASCII conversion

There is an on-going problem – whenever you convert a file from EBCDIC to ASCII and back to EBCDIC a few characters may not be properly translated. There are several solutions to this problem, but none is perfect. E2A2E should help.

I use it mainly in two situations:

- To copy a text (source) file from one system to another using a PC disk as the transport medium.
- To back-up text files on a PC.

Because I want to be able to read the files on the PC, binary transfers and file encoding are not valid options.

E2A2E is a REXX EXEC, which uses PIPES and needs VM/ESA 1.2.1 or higher.

HOW TO USE IT

Let's say that you want to copy the files ABC HELPCMS and ABC ASSEMBLE to another system.

First you create a file E2A2E LST with the lines

```
ABC HLP A HELPCMS
ABC ASM A ASSEMBLE FIXED 80
```

Now you copy to a PC disk your files as ABC.HLP and ABC.ASM, together with E2A2E.LST and E2A2E.EXC, where E2A2E.EXC is the REXX EXEC.

On the receiving system you upload all the files, rename E2A2E EXC A to E2A2E EXEC A and execute it. E2A2E will read both ABC files and re-creates ABC HELPCMS A and ABC ASSEMBLE A.

Remember, E2A2E must be subject to the same conversion as your files.

HOW IT WORKS

For E2A2E to work, it must be able to go through the conversions without corrupting the code. It was carefully written to use only ‘safe’ characters. These are the alphabetic characters plus * + - / ( ) = , and '. It also contains a positional table with all the characters to be checked. If a character was ‘lost’ during the conversions, E2A2E detects it and translates it back to the original value.

Note: To recover E2A2E enter E2A2E E2A2E EXEC A.

EXAMPLE

All files were transferred from VM to OS/2 using Communications Manager/2 and back to VM using FTP. A temporary work disk was accessed as T.

Sample console output:

```
   type e2a2e lst t

   ABC      HLP     T HELPCMS
   ABC      ASM     T ASSEMBLE FIXED 80

   Ready; T=0.01/0.01 13:23:22

   e2a2e
   Translation table...
     From:  ∫ 21 √ 3D 3C
       To:  [ 6A /  B5 B6
   File ABC HLP T converted to ABC HELPCMS T.
   File ABC ASM T converted to ABC ASSEMBLE T.
   Ready; T=0.10/0.10 13:23:32

   e2a2e e2a2e exec t
   Translation table...
     From:  ∫ 21 √ 3D 3C
       To:  [ 6A /  B5 B6
   File E2A2E EXEC T converted to E2A2E A2EEXEC T.

E2A22E A2EEXEC T is a file identical to the original E2A2E EXEC.
```

E2A2E EXEC

```
/* E2A2E - Help EBCDIC to ASCII to EBCDIC conversion                */
 Address command
```

```
/*   Special characters used by E2A2E        * + - /   ( ) =    , '   */

/*       Ø 1 2 3 4 5 6 7 8 9 A B C D E F           Character table   */
/*       . . . . . . . . . . . . . . . .                             */

chars = '  †                 [ . < ( + ]',  /* 4Ø-4F                */
        '                    ! $ * ) ; ^',  /* 5                    */
        '- /                 | , % _ > ?',  /* 6                    */
        '                    : # @ '' = "', /* 7    Quote 7D doubled */
        '  a b c d e f g h i ƒ           ',  /* 8                    */
        '  j k l m n o p q r             ',  /* 9                    */
        '— ~ s t u v w x y z             ',  /* A                    */
        'ÿ Ÿ / ¤   › fi fl ‡             ',  /* B                    */
        '{ A B C D E F G H I             ',  /* C                    */
        '} J K L M N O P Q R             ',  /* D                    */
        '\ ü S T U V W X Y Z             ',  /* E                    */
        'Ø 1 2 3 4 5 6 7 8 9             '   /* FØ-FF                */

conv  = Ø
val.  =
start = 128
nlate =                                  /* List of new (to) characters */
olate =                                  /*    old (from) characters    */
ulate =                                  /*    chars not translated     */
xlate =                                  /* PIPE xlate specification    */
flist = 'E2A2E LST *'

Do k = 1 to 512 - start by 2             /* Check character table       */
  byte = substr(chars, k, 1)             /* Build list of altered bytes */
  val  = c2x(byte)
  ind  = right(d2x((k - 1 + start) / 2), 2, 'Ø')
  If val = ind  Then Iterate

  conv = conv + 1
  ind.conv = ind                         /* Expected character (hexa)   */
  val.ind  = val
End

Do k = 1 to conv                         /* Scan list of altered bytes  */
  ind = ind.k
  val = val.ind
  If val.val = '' Then
    Select
      When x2c(val) = '' Then Iterate
      When sign(x2d(val) - start / 2) = 1 Then
        ulate = ulate ind '('x2c(ind)')'
      Otherwise
        nlate = nlate ind
        olate = olate val
```

```
        xlate = xlate val ind
    End
  Else
    Do
      nlate = nlate x2c(ind)
      olate = olate x2c(val)
      xlate = xlate val ind
    End
End

If ulate = '' Then Nop
Else Say 'Characters not recovered:' ulate

If xlate = '' Then
  Say 'No translation to be done.'
Else
  Do
    Say 'Translation table...'
    Say '  From:' olate
    Say '    To:' nlate
  End

If arg(1) = ' ' Then                  /* Check files to process      */
  Do
    list.Ø = Ø
    'ESTATE' flist
    If rc = Ø Then
      'PIPE (sep ]) filefast' flist,
          '] stem list.'
  End
Else
  Do
    list.Ø = 1
    list.1 = arg(1)
  End

If list.Ø = Ø Then                    /* Convert all files           */
  Say 'No files specified.'
Else
  Do k = 1 to list.Ø
    Parse upper var list.k fni fti fmi fto options
    expand = 'copy'
    If abbrev(word(options, 1), 'FIXED', 1) = 1 Then
      expand = 'pad' word(options, 2)
    If fto = ''  Then fto = 'A2E'fti
    If fto = fti Then fto = 'A2E'fti
    filei  = fni fti fmi
    fileo  = fni fto fmi
    'ESTATE' filei
    If rc = Ø Then
```

```
      Do
        'ESTATE' fileo
        If rc = 0 Then
          Do
            Say 'File' fileo 'already exists. Replace it (Y/N)?'
            Parse upper pull resp
            If resp = 'Y' Then
              Do
                'ERASE' fileo
                If rc = 0 Then k = k - 1
                Else Say 'Unable to erase' fileo'.'
              End
            Else
              Say 'File' filei 'not processed.'
          End
        Else
          Do
            'PIPE (sep ]) filefast' filei,
                '] xlate A A' xlate,
                ']' expand,
                '] filefast' fileo options
            If rc = 0 Then Say 'File' filei 'converted to' fileo'.'
            Else 'Error creating' fileo 'from' filei'.'
          End
      End
    Else
      Say 'File' filei 'not found.'
  End
```

E2A2E LIST

```
LESS     HLX     A HELPXEDI
PLUS     HLX     A HELPXEDI
SALT     HLX     A HELPXEDI
SWAP     HLX     A HELPXEDI
LESS     XED     A XEDIT
PLUS     XED     A XEDIT
SALT     XED     A XEDIT
SWAP     XED     A XEDIT
```

This file can be used to test the E2A2E EXEC.

Fernando Duarte
Analyst (Canada)

Backing out using 3480/3490 drives – part 2

This month we continue the system that manages back-up jobs, without using operators, so that it can be run overnight and at weekends.

PANEL TEXT

```
*START
§                        Start panel for BACKUP JOBs
!
!    BACKUP JOB name           :===>%JOBNAME !   <Required>
!
§&MSG1Ø
§&MSG11
!
!    §PFØ1!Help      §PFØ3!=Quit     §PFØ5!=Show     §PFØ6!=Rest. drives
!    §PFØ9!Def. new JOB §PF1Ø!=Def. drives §PF11!=Stop JOB
!
!
! Type a §BACKUP JOB-name! and choose a function
*END

             ---------------------------------------------

*START1
§           Panel for defining tape drives to a BACKUP JOB.
!
!     START tape   :===>%S.1 !  %S.2 !  %S.3 !  %S.4 !   <Max. 4 drives
!                  :===>%S.5 !  %S.6 !  %S.7 !  %S.8 !    to each line>
!                  :===>%S.9 !  %S.1Ø!  %S.11!  %S.12!
!                  :===>%S.13!  %S.14!
!
!       DRAIN tape  :===>%D.1 !  %D.2 !  %D.3 !          <Max. 3 drives
!                   :===>%D.4 !  %D.5 !  %D.6 !           to each line>
!                   :===>%D.7 !  %D.8 !  %D.9 !
!                   :===>%D.1Ø!  %D.11!  %D.12!
!                   :===>%D.13!
§&MSG2Ø
§&MSG21
!
!
!   §PFØ3!=Main menu                              §PFØ5!=Process
!                    §PF11!=Panel for stop date
!
!
```

```
! Type addresses for drives to be used in the §&NAME    ! BACKUP JOB
! Remember also to define drives that are not to be used for this JOB
*END1

             ------------------------------------------------

*START2
§      Panel for defining dates when a BACKUP JOB is not to be performed
!
!                                              DAY   MNT  YEAR    CMNT
!          Skip JOB (date: dd mm yy   :===>%D1!  %M1!  %Y1!    %C1!
!                                     :===>%D2!  %M2!  %Y2!    %C2!
!                                     :===>%D3!  %M3!  %Y3!    %C3!
!                                     :===>%D4!  %M4!  %Y4!    %C4!
!                                     :===>%D5!  %M5!  %Y5!    %C5!
!                                     :===>%D6!  %M6!  %Y6!    %C6!
!                                     :===>%D7!  %M7!  %Y7!    %C7!
!                                     :===>%D8!  %M8!  %Y8!    %C8!
§&MSG3Ø
§&MSG31
!
!
!        §PFØ3!=Main Menu                       §PFØ5!=Process
!
!
   Type date(s) for possible stop(s) of §&NAME    ! BACKUP
!
§           If you want to remove a date from the list:
§           Type the date and also RE in the CMNT field
!
*END2

               ----------------------------------------

*START3
§                   Display  parameters defined for
!                             §&MSG4Ø
!
§&MSG4.1
§&MSG4.2
!
§&MSG4.3
§&MSG4.4
!
§&MSG4.5
§&MSG4.6
§&MSG4.7
§&MSG4.8
§&MSG4.9
```

```
§&MSG4.1Ø
§&MSG4.11
!
! Lines marked  §*! kan be retrieved (reactivated)
! by choosing §Restore! from the Main menu.
!
!
!                                §PFØ3!=Main menu
!                                §CLEAR!=Show more
*END3
```

```
*START4

This BACKUP system consists of a set of EXECs and other files that can
be used to simplify daily BACKUP routines.

A scheduling system (VMSCHED) starts a BACKUP job at a predefined time.
The EXEC that corresponds to the job uses some additional EXECS (SKIP,
DRAINSTA, TAPDRAIN, and TAPSTART) to withdraw parameters pertinent to
the job, from the BACKUP PARMS file. If the retrieved parameters say
that the job should be run, then other data from the file will tell
which drives should be started and which should be drained before the
job can start. When tape drives have been set up, a message is sent to
VMBACKUP to start the job.

The BACKUP PARMS file has four different recordtypes. All records starts
with a jobname (ie MONTHLY DAILY etc). The next field on the record
contains the words JOB, START, DRAIN, or SKIP. There must be only one
'space' between the jobname and the second parameter.

The record 'jobname JOB' is a mandatory record. This record must be
defined before other records for the job can be entered. All records can
be defined with the aid of the BACKUP EXEC which performs extensive data
checking.

The record 'jobname START' defines which tape drives VMTAPE should start
for the job. The format of this record is:
Jobname START 88Ø 881 882 .......

!     §PFKØ3!=Main menue           §PFKØ8!=Forward
*END4
```

```
*START5

The record 'jobname DRAIN' defines which tape drives VMTAPE should drain
```

```
for the job. The format of this record is:
Jobname DRAIN 883 884 885 .......

The record 'jobname SKIP' tells the system that this job should be
skipped (not performed) for the date specified. The format is: jobname
SKIP 19yymmdd. There is no limit to the number of SKIP records that can
be entered for a job.

When the operator uses the BACKUP EXEC to change the START and DRAIN
records for a job, then the old definitions will be kept, but now with a
'*' in front of the jobname. If this 'old' definition should be made
active again, it can be done from the main menu through the choice
'Rest. drives'. Only one set of '*jobname' records can be kept. If the
need should arise that an additional definition has to be made, the old
'*jobname START' and '*jobname DRAIN' have to be removed manually. Any
blank lines introduced to the file will be removed by the same EXEC that
removes obsolete SKIP records (REMVSKIP). This EXEC should
be run once a day from the same userid that keeps the BACKUP PARMS file.

The system can run several jobs in 'parallel' provided that enough time
is allowed for, between start-ups, so that the drives to be used have
been attached to the system performing the back-up (ie VMBACKUP).


!       §PFKØ3!=Main menu               §PFKØ7!=Backwards



*END5

             ------------------------------------------
```

BKPMAINT TEXT

```
/*                                                                    */
/**********************************************************************/
/*                                                                    */
/*   Starts maintenance of back-up jobs on user-id BACKUP             */
/*                                                                    */
/**********************************************************************/
/*                                                                    */
address command
rk = diagrc('8','LINK BACKUP 191 333 MR')
parse var rk rc .
if rc = Ø then
'ACCESS 333 M'
else
if rc = 298 then
```

```
do
  say 'You have no permission to link BACKUPs 191 disk'
  exit
end
else
do
  say 'BACKUPs 191 disk is linked RW by another user, try again later'
  rk = diag('8','DET 333')
  exit
end
rk = diagrc('8','LINK BACKUP 192 334 MR')
parse var rk rc .
if rc = Ø then
'ACCESS 334 N'
else
if rc = 298 then
do
  say 'You have no permission to link BACKUPs 192 disk'
  'RELEASE M'
  rk = diag('8','DET 333')
  exit
end
else
do
  say 'BACKUPs 192 disk is linked RW by another user, try again later'
  rk = diag('8','DET 334')
  exit
end
'EXEC BACKUP'
'RELEASE M'
'RELEASE N'
 rk = diag('8','DET 333')
 rk = diag('8','DET 334')
 exit
```

E21RES TEXT

```
/*                                                                    */
/**********************************************************************/
/*                                                                    */
/* Set up a DDR back-up job. The tapes to be used will be mounted     */
/* and catalogued by VMTAPE.                                          */
/* This EXEC shold be run on a specific userid since it must be       */
/* active as long as the DDR is.                                      */
/*                                                                    */
/**********************************************************************/
/*                                                                    */
Arg fn ft fm
```

```
'SET CMSTYPE HT'
if userid() = 'BACKUP' then 'CP LOGOFF'
'EXEC SKIP E21RES'
'SPOOL CONS * CL U START'
'LINK MAINT 123 123  RR '    /* MAINT fullpack DIRECTORY definition   */
status = 'DSC'
/*                                                                    */
/**********************************************************************/
/*                                                                    */
/* Test if BACKUP is logged on (Disconnected).                        */
/*                                                                    */
/**********************************************************************/
/*                                                                    */
do while status = 'DSC'   /* Test if BACKUP is DSC (not ready)        */
  'SLEEP 1 SEC'
  'EXECIO * CP (STRING QUERY BACKUP'
   parse pull . . status .
end
/*                                                                    */
/**********************************************************************/
/*                                                                    */
/* BACKUP is now ready to accept a new job. (ie logged off)           */
/* Ask BACKUP to set up tape drives for this job.                     */
/*                                                                    */
/**********************************************************************/
/*                                                                    */
'CP AUTOLOG BACKUP DRAINSTA E21RES'
/*                                                                    */
/**********************************************************************/
/*                                                                    */
/* Wait while BACKUP performs what it is told to do.                  */
/* BACKUP should only be running for a few seconds.                   */
/*                                                                    */
/**********************************************************************/
/*                                                                    */
'SLEEP 60 SEC'
/*                                                                    */
/**********************************************************************/
/*                                                                    */
/* Ask VMTAPE to mount a tape. When mounted, start DDR back-up.       */
/*                                                                    */
/**********************************************************************/
/*                                                                    */
'SET CMSTYPE RT'
'VMTAPE MOUNT SCRATCH 181 DSN E21RES (WAIT LABEL SL UNIT 3480 RET 30'
'NUCXLOAD WRITLINE ( SYSTEM'
'XDDR E21RES DDR *'
'NUCXDROP WRITLINE'
'DETACH 181'
```

```
'SPOOL CONS CLOSE STOP'
'EXEC TRANSLOG E21RES' userid()  /* Transfer the log to rec. userid  */
'LOGOFF'
```

DDR TEXT

```
SY CONS
IN 123 338Ø E21RES
OUT 181 348Ø (SKIP 1 UNLOAD COMPACT
DU ALL
```

MONTHLY TEXT

```
/*                                                                  */
/********************************************************************/
/*                                                                  */
/* Set up a monthly back-up job for execution.                      */
/*                                                                  */
/********************************************************************/
/*                                                                  */
'EXEC SKIP MONTHLY'           /* Test if MONTLY job should be skipped */
'EXEC STADRAIN MONTHLY'       /* START/DRAIN necessary tape drives    */
'VMBACKUP START MONTHLY'      /* Start the job                        */
'LOGOFF'
```

SKIP TEXT

```
/*                                                                  */
/********************************************************************/
/*                                                                  */
/* Tests BACKUP PARMS for possible skip of a back-up job.  If the job */
/* should be skipped then BACKUP is logged off.                     */
/* Input to SKIP is jobname, ie SKIP monthly.                       */
/*                                                                  */
/********************************************************************/
/*                                                                  */
arg jobname
date = date(S)
'pipe',
'< BACKUP PARMS *',
'! stem parms.'
do k = 1 to parms.Ø
  if word(parms.k,1) = jobnavn & word(parms.k,2) = 'SKIP' ,
                               & word(parms.k,3) = date then 'LOGOFF'
end
exit
```

STADRAIN TEXT

```
/*                                                                    */
/**********************************************************************/
/*                                                                    */
/* Starts or drains tape stations according to information located in */
/* BACKUP PARMS file.                                                 */
/* Input to STADRAIN is BACKUP jobnavn, ie STADRAIN daily             */
/*                                                                    */
/**********************************************************************/
/*                                                                    */
arg jobname
dato = date(S)
'pipe',
'< BACKUP PARMS *',
'! stem parms.'
do k = 1 to parms.Ø                    /* Check BACKUP PARMS file     */
  if word(parms.k,1) = jobnavn then
  do
    if word(parms.k,2) = 'START' then cmd1 = 'EXEC TAPSTART'
subword(parms.k,3)
    if word(parms.k,2) = 'DRAIN' then cmd2 = 'EXEC TAPDRAIN'
subword(parms.k,3)
  end
end
cmd2
'SLEEP 1Ø SEC'
cmd1
'CP LOGOFF'
```

TAPSTART TEXT

```
/*                                                                    */
/**********************************************************************/
/*                                                                    */
/* TAPSTART starts requested tape drives. It also tests if users other*/
/* than predefined have the specified drive attached. If so the tape  */
/* drive is DETACHed. VMTAPE is then asked to START the drive.        */
/*                                                                    */
/**********************************************************************/
/*                                                                    */
arg string
address command
'MAKEBUF'
'VMTAPE QUERY TAPES (STACK'
do while queued() > Ø
  pull line
  parse var line . unit type . status user .
```

```
  if type = '18,348Ø' then
  do
    if pos(unit,string) > Ø then
    do
      if status = 'ATTACHED' & (user ¬= 'VMBACKUP' & user ¬= 'BACKUP',
                  & user ¬= 'BACKUP1' & user ¬= 'BACKUP2',
                  & user ¬= 'BACKUP3' & user ¬= 'BACKUP4',
                  & user ¬= 'BACKUP5' ) then
      do
        'CP DETACH' unit 'FROM'  user 'LEAVE' /* Take drive from user */
        'VMTAPE START' unit
      end
      if status = 'DRAINED' then
      do
        'VMTAPE START' unit
      end
    end
  end
end
'DROPBUF'
```

TAPDRAIN TEXT

```
/*                                                                    */
/**********************************************************************/
/*                                                                    */
/* Tape drives are DRAINed according to information contained in the  */
/* input string and the status of the drives.                         */
/*                                                                    */
/**********************************************************************/
/*                                                                    */
arg string
'MAKEBUF'
'VMTAPE QUERY TAPES (STACK'
do while queued() > Ø               /* Test status for tape drives    */
  pull line
  parse var line . unit type . status user .
  if pos(unit,string) > Ø then
  do
    if status = 'FREE' then
    do
      'VMTAPE DRAIN' unit
    end
  end
end
'DROPBUF'
```

TRANSLOG TEXT

```
/*                                                                    */
/**********************************************************************/
/*                                                                    */
/*Receives the log from VMBACKUP to disk. Renames it according to     */
/*contents and forwards the log to a specified user-id. In this case  */
/*to VMBSYSAD                                                         */
/*                                                                    */
/**********************************************************************/
/*                                                                    */
arg job userid
'MAKEBUF'
'EXECIO * CP (STEM Q_RDR. STRING Q RDR * ALL'
numb = q_rdr.Ø
do i = 1 to numb
  parse var q_rdr.i orig spoolid C type . . . date time .
  if type = 'CON' then
  do
    if orig = userid & C = 'U' then
    do
      'RECEIVE' spoolid job 'SYSLOG A'
      'EXECIO * DISKR' job 'SYSLOG A (LOCATE /DUMPING DATA/'
      if rc = Ø then
      do
        pull line
        pull line
        'FINIS' job 'SYSLOG A'
        word = word(line,8)
        if word = job then
        do
          'SENDFILE' job 'SYSLOG A TO VMBSYSAD'
          'ERASE' job 'SYSLOG A'
        end
        else
        do
          'RENAME' job 'SYSLOG A' word '= ='
          'SENDFILE' word 'SYSLOG A TO VMBSYSAD'
          'ERASE' word 'SYSLOG A'
        end
      end
      else
        'ERASE' job 'SYSLOG A'
    end
  end
  if c = 'T' then 'CP PURGE RDR' spoolid
end
'DROPBUF'
```

REMVSKIP TEXT

```
/*                                                                  */
/********************************************************************/
/*                                                                  */
/* Removes all skipdates from BACKUP PARMS file which are           */
/* timestamped earlier than today's date.                           */
/*                                                                  */
/********************************************************************/
/*                                                                  */
date = date(s)
'pipe',
'< BACKUP PARMS A',
'! stem parms.'
l = 0
do k = 1 to parms.0
  if word(parms.k,2) = 'SKIP' then
  do
    if word(parms.k,3) >= date then
    do
      l = l + 1
      parms_new.0 = l
      parms_new.l = parms.k
    end
  end
  else
  if line ¬= '' then
  do
    l = l + 1
    parms_new.0 = l
    parms_new.l = parms.k
  end
end
'pipe',
'stem parms_new.',
'! > BACKUP PARMS A F 80'
'LOGOFF'
```

RESTADDR EXEC

```
/*                                                                  */
/********************************************************************/
/*                                                                  */
/*  RESTADDR EXEC (RESTore Automatic DDR)                           */
/*  Sample EXEC for use while restoring a DDR with automatic tape   */
/*  mounting using tape handling software                           */
/*                                                                  */
/********************************************************************/
```

```
/*                                                                    */
arg  ddr_file tape_list
parse var tape_list  first_tape rest_tapes
'NUCXDROP WRITLINE'
'NUCXLOAD WRITLINE (SYSTEM'
'VMTAPE MOUNT' first_tape ' 181 (WAIT LABEL SL'
if rest_tapes ¬= '' then 'WRITLINE MOUNT' rest_tapes
'XDDR' ddr_file 'DDR *'
'NUCXDROP WRITLINE'
```

RESTMDDR EXEC

```
/*                                                                    */
/**********************************************************************/
/*                                                                    */
/*  RESTMDDR EXEC (RESTore Manual DDR)                                */
/*  Sample EXEC for use while restoring a DDR using manual tape       */
/*  mounting.                                                         */
/*                                                                    */
/**********************************************************************/
/*                                                                    */
arg  ddr_file
'NUCXDROP WRITLINE'
'NUCXLOAD WRITLINE (SYSTEM'
'XDDR' ddr_file 'DDR *'
'NUCXDROP WRITLINE'
```

Odd Hatlevold
Senior Systems Programmer
Statoil (Norway)

Subscribers who want copies of the code from this issue can call our Web site – www.xephon.com – and ask for the article they require. The article will then be e-mailed to them. This service is free to subscribers. Subscribers will need their user-id (which is on the mailing label on the envelope containing this issue), and they will need a copy of this issue so that they can answer a simple question (this is to prevent non-subscribes accessing information that subscribers have paid for).

Improving SQL/DS programming technology

The application programming technology determines how the resources of an information centre are used. This covers both the cost of the information services offered and the number of production databases that may be maintained concurrently.

The new technology of SQL/DS application programming optimizes the use of the following important resources:

- The time needed for the creation and debugging of new SQL/DS production modules.
- SQL/DS virtual machine performance during SQL/DS program development.

The modules, which support the new SQL/DS application programming technology, are written in Assembler and REXX.

The program code is developed in CMS with SQL/DS Version 2.1.

NEW TECHNOLOGY SPECIFICATION

As a rule, the debugging of SQL/DS working modules requires repeated translations to be carried out before the final results are obtained. Prior to each translation, the SQL/DS preprocessor must be called, to replace SQL/DS statements with host language code – see Figure 1.

In the most common cases, the host language code is the subject of change, not the embedded SQL/DS statements. The additional SQL/DS preprocessor calls, when the SQL/DS source code is not changed, are not a complete waste of time because of possible locking and unexpected growth of the total translation time. As a result of putting this conventional technology into practice, the following negative consequences result:

- Bad response time of SQL/DS caused by unnecessary activity of the SQL/DS virtual machine and locking in system dbspaces.

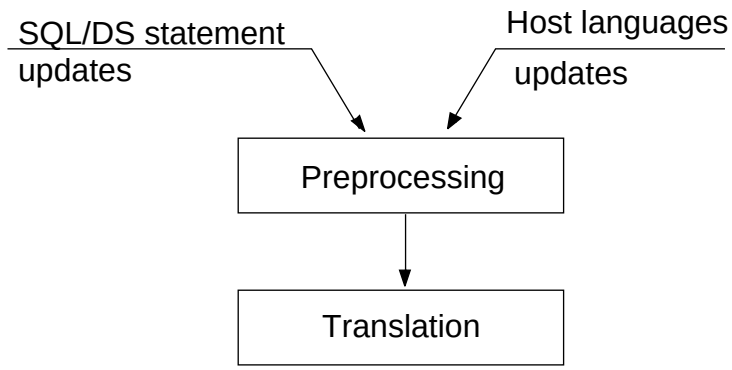

Figure 1: Conventional SQL/DS application development

- Bad response time of VM/CMS because of system loading, causing a high system paging rate.

The new SQL/DS program development technology is based on the use of REXX and Assembler code, which improve performance, after successfully preprocessing the host language statements. In future, these statements can be included in the debugging program, without needing to call the SQL/DS preprocessor – see Figure 2.

Application programmers can now make changes to program code and debug without preprocessing. This allows programmers to do their work more quickly and more efficiently. System resources are released and performance is improved. As a result the cost of information services goes down.

PREPT USAGE

PREPT EXEC implements the improved SQL/DS programming technology. It is used to preprocess and translate or only translate the

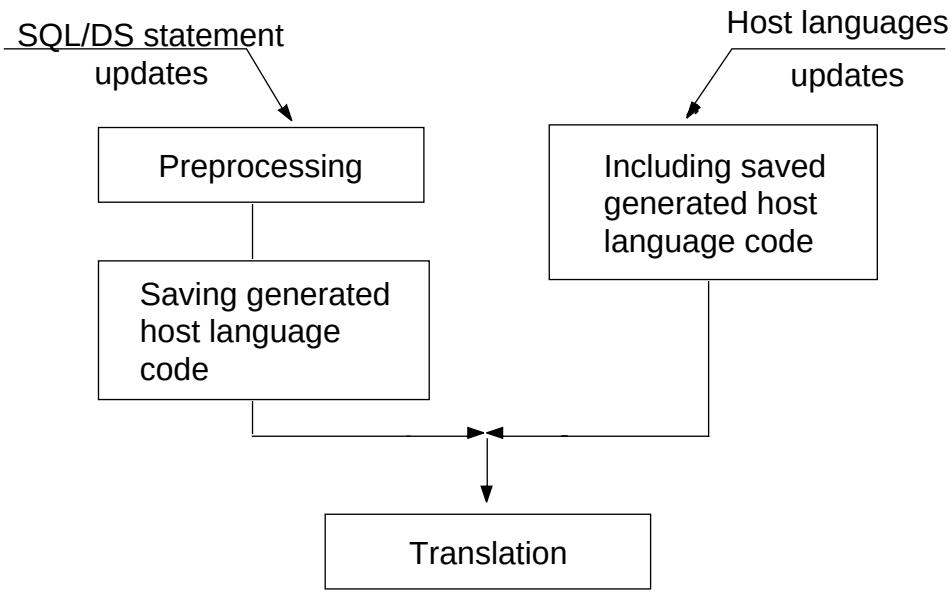

Figure 2: New SQL/DS application development technology

SQL/DS applications programs written in PL/I, C, and COBOL host programming languages.

The PREPT EXEC is invoked as follows:

```
PREPT <prep_call> <host_lang> <macro_name> [(<options>]
```

where:

- prep_call – if P is used, it identifies the SQL/DS preprocessor to be called. Any other non-blank character cancels application program preprocessing.
- host_lang – this may be PLI, C, or COB. It identifies the host programming language.
- macro_name – this is the filename of a file with the filetype MACRO on mini-disk A, where the source program code is located.

- options – may contain options for the corresponding compiler.

Do not delete or modify the following files with XEDIT:

- <macro_name> _SQLGEN_ A
- <macro_name> _SQLGEN1 A.

These files contain statements, generated by the SQL/DS preprocessor for use when program preprocessing is cancelled, ie <prep_call> is not P. Use PREPT EXEC to restore preprocessor generated code from them.

The following restrictions must be considered when working with PREPT EXEC:

- The preprocessor keyword EXEC should be located in the first 20 positions of the source record.
- SQL/DS statements must not contain comments inside themselves.
- Source program files must be serialized, if they are not already.

  This can done by the XEDIT command:

  ```
  SER ON
  ```

  It is handy to put this command in PROFILE XEDIT.

Examples of PREPT EXEC are:

- Preprocessing and translating PLPROG MACRO A written in PL/I:

  ```
  PREPT P PLI PLPROG
  ```

- Translating only CPROG MACRO A, written in C:

  ```
  PREPT N C CPROG
  ```

- Translating only COBPROG MACRO A written in COBOL and using a special parameter for the COBOL compiler:

  ```
  PREPT N COB COBPROG (APOST
  ```

INSTALL EXEC

```
/*****************************************************************/
/***                                                  ***         ***/
/*** INSTALL          generate INCLSQL MODULE         ***  DG'97  ***/
/***                                                  ***         ***/
/*****************************************************************/
/***   SIZE 00035  VER 1.0 MOD 00  TIME 16:12:43  DATE 20/06/97   ***/
/*****************************************************************/

  CLRSCRN
  MESSAGE = 'user request'
  SAY ' --- Start INCLSQL MODULE generation - reply Y or N'
  PULL REPLY
  IF REPLY ¬= 'Y' THEN
  SIGNAL ERROR
  SET CMSTYPE HT
  SIGNAL ON ERROR
  MESSAGE = 'error when assemble' INCLSQL
  ASSEMBLE INCLSQL
  ERASE INCLSQL LISTING A
  MESSAGE = 'error when assemble' EXPAND
  ASSEMBLE EXPAND
  ERASE EXPAND LISTING A
  MESSAGE = 'error when load' INCLSQL
  LOAD INCLSQL '(' NOMAP NOLIBE AUTO RLDSAVE
  MESSAGE = 'error when genmod' INCLSQL
  GENMOD
  ERASE INCLSQL TEXT A
  ERASE EXPAND TEXT A
  SIGNAL OFF ERROR
  SET CMSTYPE RT
  SAY ' --- INCLSQL MODULE generated successfully'
  EXIT
ERROR:
  SET CMSTYPE RT
  SAY ' --- INCLSQL MODULE not generated due to' MESSAGE
```

EXPAND ASSEMBLE

```
*********************************************************************
****                                                 ***         ****
**** EXPAND      expand source from lookat record    ***  DG'97  ****
****                                                 ***         ****
*********************************************************************
****   SIZE 00107  VER 1.0 MOD 00  TIME 15:50:27  DATE 20/06/97   ****
*********************************************************************
*                                                                   *
EXPAND   CSECT
```

```
         PRINT NOGEN
         SAVE  (14,12)
         BALR  2,Ø
         USING *,2
         LR    3,1
         USING PARMS,3
         L     15,LOOKAT
         LH    15,Ø(15)
         L     4,SIZEOF
         LH    14,Ø(4)
         L     5,BUFSIZE
         LH    5,Ø(5)
         L     6,TOEXPAND
         LH    6,Ø(6)
         LR    Ø,6
         LTR   Ø,Ø
         BZ    RET
         LR    1,Ø
         AR    1,14
         STH   1,Ø(4)
         CR    1,5
         BH    NOSPACE
         LR    1,Ø
         LR    11,15
         LR    12,15
         AR    12,Ø
         SR    14,15
         LA    14,1(14)
         LR    Ø,14
         BAL   4,MOVEIT
         LR    11,15
         LR    12,15
         LR    Ø,1
         BAL   4,MOVEIT
RET      EQU   *
         RETURN (14,12)
NOSPACE  EQU   *
         XC    Ø(2,4),Ø(4)
         B     RET
MOVEIT   EQU   *
         MH    Ø,=H'8Ø'
         MH    11,=H'8Ø'
         MH    12,=H'8Ø'
         L     6,BUFADDR
         L     6,Ø(6)
         AR    11,6
         AR    12,6
         LA    14,MOVEREST
         LR    1Ø,12
```

```
         SR    1Ø,11
         LR    9,1Ø
         LPR   1Ø,1Ø
         BZ    SETOFF
         CR    Ø,1Ø
         BH    MODIFY
SETOFF   EQU   *
         LR    1Ø,Ø
         B     ONETIME
MODIFY   EQU   *
         LTR   9,9
         BM    ONETIME
         AR    11,Ø
         AR    12,Ø
         LA    14,NEGATIVE
NEGATIVE EQU   *
         SR    11,1Ø
         SR    12,1Ø
ONETIME  EQU   *
         LR    6,12
         LR    8,11
MOVEREST EQU   *
         LR    7,1Ø
         LPR   7,7
         CR    6,8
         BNE   SETLEN
         SR    9,9
         ICM   9,8,=X'4Ø'
         B     EXECMVCL
SETLEN   EQU   *
         LR    9,7
EXECMVCL EQU   *
         MVCL  6,8
         SR    Ø,1Ø
         LTR   Ø,Ø
         BNZ   MODLEN
         BR    4
MODLEN   EQU   *
         CR    Ø,1Ø
         BCR   2,14
         LR    1Ø,Ø
         BR    14
PARMS    DSECT
TOEXPAND DS    A
BUFADDR  DS    A
BUFSIZE  DS    A
LOOKAT   DS    A
SIZEOF   DS    A
         END   EXPAND
```

INCLSQL ASSEMBLE

```
*********************************************************************
****                                                     ***         ****
**** INCLSQL      include SQL/DS extra statements       ***  DG'97  ****
****                                                     ***         ****
*********************************************************************
****   SIZE ØØ245  VER 1.Ø MOD Ø2  TIME 16:54:42  DATE 2Ø/Ø6/97   ****
*********************************************************************
*                                                                   *
INCLSQL  CSECT
         PRINT NOGEN
         USING *,12
         LR    1Ø,14
         SR    15,15
         ICM   15,3,SETLANG+4
         CLC   8(3,1),=C'ASM'
         BNE   CHECKC
         MVC   ZAP(ASMILEN),ASMI
         MVC   ZAP2(ASMELEN),ASME
         B     AFORTZAP
CHECKC   EQU   *
         CLC   8(2,1),=CL2'C'
         BNE   CHECKCOB
         MVI   CPLI1+3,X'ØØ'
         MVC   JUMPONE(4),=2X'Ø7ØØ'
         LA    15,16(15)
         B     CPLIZAP
CHECKCOB EQU   *
         CLC   8(3,1),=C'COB'
         BNE   CHECKFOR
         MVC   ZAP1(L'COB),COB
         MVC   ZAP3(L'COBE),COBE
         MVC   ZAP4(L'COB),COB
         MVC   ZAP(L'COBI),COBI
         MVC   JUMPTWO(4),=2X'Ø7ØØ'
         LA    15,8(15)
         B     PROCIT
CHECKFOR EQU   *
         CLC   8(4,1),=C'FORT'
         BNE   CHECKPLI
         MVC   ZAP(L'FORTI),FORTI
         MVC   ZAP2(FORTELEN),FORTE
AFORTZAP EQU   *
         MVC   ZAP1(L'ASMFORT),ASMFORT
         MVC   ZAP4(L'ASMFORT),ASMFORT
         B     PROCIT
CHECKPLI EQU   *
```

```
         CLC   8(3,1),=C'PLI'
         BNE   WRNGPRM
CPLIZAP  EQU   *
         MVC   ZAP1(L'CPLI1),CPLI1
         MVC   ZAP3(L'CPLIE),CPLIE
         MVC   ZAP4(L'CPLI2),CPLI2
PROCIT   EQU   *
         MVC   FNAME(18),16(1)
         MVC   SNAME(8),16(1)
         STH   15,SETLANG+4
         DMSFREE DWORDS=100000,ERR=NOTMEM,AREA=HIGH
         LR    2,1
         ST    2,BUFADDR
         LA    4,FNAME
         FSOPEN  (4)
         LTR   15,15
         BNE   NOTFOUND
         FSREAD (4),NOREC=7000,BSIZE=560000,BUFFER=(2)
         LTR   15,15
         BNZ   READERR
         LR    14,0
         SRDL  14,32
         D     14,=F'80'
         STH   15,SIZEOF
         LR    8,0
         BCTR  0,0
         LR    3,2
         AR    3,0
         CH    15,=H'7000'
         BL    CLOSE
         FSREAD (4),NOREC=1,BSIZE=80,BUFFER=REC
         CH    15,=H'12'
         BNE   BIGFILE
CLOSE    EQU   *
         FSCLOSE (4)
         LR    1,2
         LA    2,80
         SR    11,11
CYC      EQU   *
         LA    11,1(11)
         CLI   SQL,C'N'
         BE    MISS
ZAP      DC    4X'0700'
MISS     EQU   *
         LR    5,1
         LA    6,1
         LA    7,16(1)
         CLC   0(5,5),=CL5'EXEC'
         BE    SETON
```

```
CHECKSQL EQU   *
         CLC   Ø(6,5),=CL6' EXEC'
         BE    SETON
         BXLE  5,6,CHECKSQL
         B     CHECKSET
SETON    EQU   *
         MVI   SQL,C'Y'
ZAP1     MVC   1(2,1),=C'/*'
CHECKSET EQU   *
         CLI   SQL,C'N'
         BE    CONTINUE
ZAP2     DC    6X'Ø7ØØ'
         LR    5,1
         LA    7,71(1)
CHECKEND EQU   *
ZAP3     CLC   Ø(1,5),=C';'
         BE    SETOFF
         BXLE  5,6,CHECKEND
         B     CONTINUE
SETOFF   EQU   *
ZAP4     MVC   69(2,1),=C'*/'
         MVI   SQL,C'N'
INCLUDE  EQU   *
         STM   13,5,RG135
         LA    4,8Ø(1)
         LA    5,SNAME
         FSREAD  (5),BSIZE=8Ø,BUFFER=REC,NOREC=1
         LTR   15,15
         BNZ   UNKNOWN
         PACK  DOUBLE(8),REC(4)
         NI    DOUBLE+7,X'FØ'
         OI    DOUBLE+7,X'ØC'
         CVB   2,DOUBLE
         STH   2,TOEXPAND
         LTR   2,2
         BZ    GETBACK
         STH   11,LOOKAT
         LA    13,SAVRTBGN
         CALL  EXPAND,(TOEXPAND,BUFADDR,BUFSIZE,LOOKAT,SIZEOF)
         CLC   SIZEOF,=2X'ØØ'
         BE    NOTMEMRY
         AH    11,TOEXPAND
         LA    3,8Ø
         MH    3,TOEXPAND
         ST    3,SA
         FSREAD  (5),BSIZE=(3),BUFFER=(4),NOREC=(2)
         LTR   15,15
         BNZ   UNKNOWN
         LM    13,5,RG135
```

```
         A     1,SA
         A     3,SA
WHRETOGO B     CONTINUE
GETBACK  EQU   *
         LM    13,5,RG135
CONTINUE EQU   *
         BXLE  1,2,CYC
JUMPONE  B     JUMPTWO
         MVI   SNAME+15,C'1'
         MVC   WHRETOGO,JUMPEND
         SR    11,11
         L     1,BUFADDR
         SH    1,=H'8Ø'
         B     INCLUDE
JUMPTWO  B     JUMPDONE
         MVI   SNAME+15,C'1'
         MVC   WHRETOGO,JUMPEND
         SR    11,11
         L     1,BUFADDR
         LH    3,SIZEOF
         BCTR  3,Ø
         MH    3,=H'8Ø'
         AR    3,1
COBONLY  EQU   *
         LA    11,1(11)
         CLC   7(4,1),=CL4'PROC'
         BE    INCLUDE
         BXLE  1,2,COBONLY
JUMPDONE EQU   *
         LH    3,SIZEOF
         LR    8,3
         MH    8,=H'8Ø'
         L     11,BUFADDR
SETLANG  MVC   8(8,4),LANG+Ø
         FSWRITE (4),BSIZE=(8),BUFFER=(11),NOREC=(3),RECNO=1
         LTR   15,15
         BNZ   WRITERR
         FSCLOSE (4)
FREEMEM  EQU   *
         L     1,BUFADDR
         DMSFRET DWORDS=1ØØØØØ,LOC=(1)
RETURN   EQU   *
         BR    1Ø
DOUBLE   DS    D
SAVRTBGN EQU   *-12
SA       DS    15F
RG135    DS    9F
TOEXPAND DS    H
BUFADDR  DS    F
```

```
BUFSIZE  DC    H'10000'
LOOKAT   DS    H
SIZEOF   DS    H
JUMPEND  B     JUMPDONE
ASMFORT  MVC   0(1,1),=C'*'
ASMI     MVI   0(1),C'*'
         MVI   71(1),X'40'
ASMILEN  EQU   *-ASMI
ASME     CLI   71(1),X'40'
         BE    CONTINUE
         B     SETOFF
ASMELEN  EQU   *-ASME
FORTI    MVI   0(1),X'40'
FORTE    CLI   5(1),X'40'
         BE    CONTINUE
         B     SETOFF
FORTELEN EQU   *-FORTE
CPLI1    MVC   1(2,1),=C'/*'
CPLI2    MVC   69(2,1),=C'*/'
CPLIE    CLC   0(1,5),=C';'
COB      MVC   6(1,1),=C'*'
COBE     CLC   0(8,5),=C'END-EXEC'
COBI     MVI   6(1),C'*'
WRNGPRM  EQU   *
         LINEDIT TEXT='--- Wrong parameters',DOT=NO
         B     RETURN
NOTMEM   EQU   *
         LINEDIT TEXT='--- Not enough memory to allocate',DOT=NO
         B     RETURN
NOTFOUND EQU   *
         LINEDIT TEXT='--- File not found',DOT=NO
         B     FREEMEM
READERR  EQU   *
         LINEDIT TEXT='--- Read I/O error',DOT=NO
         B     FREEMEM
BIGFILE  EQU   *
         LINEDIT TEXT='--- More than 7000 lines to read',DOT=NO
         B     FREEMEM
WRITERR  EQU   *
         LINEDIT TEXT='--- Write I/O error',DOT=NO
         B     FREEMEM
UNKNOWN  EQU   *
         LINEDIT TEXT='--- Unknown file <filename> _SQLGEN_ A',DOT=NO
         B     FREEMEM
NOTMEMRY EQU   *
         LINEDIT TEXT='--- More than 10000 lines to generate',DOT=NO
         B     FREEMEM
FNAME    DS    18C
SNAME    DC    CL18'12345678_SQLGEN_A'
```

```
LANG     DC    CL24'PLIOPT  COBOL   C'
REC      DS    8ØC
SQL      DC    C'N'
         END   INCLSQL
```

EXTRSQL EXEC

```
/*********************************************************************/
/***                                                    ***         ***/
/*** EXTRSQL      extract SQL/DS extra statements        ***  DG'97  ***/
/***                                                    ***         ***/
/*********************************************************************/
/***   SIZE ØØØ86  VER 1.Ø MOD Ø1  TIME 16:21:33  DATE 2Ø/Ø6/97   ***/
/*********************************************************************/

  PARSE ARG FN FT FM

  SET CMSTYPE HT
  STATE FN FT FM
  IF RC ¬= Ø THEN
  DO
    SET CMSTYPE RT
    SAY '>>>--->' FN FT FM 'not found'
    EXIT
  END
  ERASE FN _SQLGEN_ FM
  ERASE FN _SQLGEN1 FM
  SET CMSTYPE RT
  REC_NO_START = 1
  LAST_W = Ø
  LAST_EXTRA_REC = Ø
  STATIC_DATA = 1
  DO FOREVER
   'EXECIO * DISKR' FN FT FM REC_NO_START '(FI / / Z 73 73 LIFO'
    IF RC ¬= Ø THEN
    LEAVE
    PULL . REC_NO_START
    PULL
   'EXECIO * DISKR' FN FT FM '(A / / Z 73 73 LIFO'
    PULL . REC_NO_END
    PULL
    IF REC_NO_END - REC_NO_START < 3 THEN
    DO
     'EXECIO' 2 'DISKR' FN FT FM REC_NO_END '(LO /XEC / Z 2 2Ø SK'
      IF RC = Ø THEN
      DO
       'EXECIO * DISKR' FN FT FM REC_NO_END '(FI / / Z 73 73 LIFO SK'
       'EXECIO * DISKR' FN FT FM '(A / / Z 73 73 LIFO'
```

```
      PULL . REC_NO_END
      PULL
    END
  END
  IS_SQL = 'Y'
  DO WHILE (IS_SQL = 'Y')
    REC_NO_SQL = REC_NO_END
    EXTRA_REC = REC_NO_SQL - REC_NO_START
   'EXECIO' EXTRA_REC                                                  ,
            DISKR FN FT FM REC_NO_START '(LO /XEC / Z 2 2Ø'
    IF RC ¬= Ø  THEN
    DO
     'EXECIO 1 DISKW' FN '_SQLGEN'STATIC_DATA'_' FM Ø F                 ,
              '(ST' RIGHT(EXTRA_REC, 4, 'Ø')  REC_NO_START
     'EXECIO' EXTRA_REC 'DISKR' FN FT FM REC_NO_START
     'EXECIO' EXTRA_REC 'DISKW' FN '_SQLGEN'STATIC_DATA'_' FM
      STATIC_DATA = STATIC_DATA + 1
      IS_SQL = 'N'
    END
    ELSE
    DO
      PULL . MID_END
      PULL
      IF EXTRA_REC > 2 THEN
      DO
       'EXECIO' REC_NO_END - MID_END - 1                                ,
                DISKR FN FT FM MID_END + 1 '(LO /XEC / Z 2 2Ø'
        IF RC = Ø THEN
        DO
          PULL . REC_NO_SQL
          PULL
          EXTRA_REC = REC_NO_SQL - MID_END
          REC_NO_SQL = REC_NO_SQL - 1;
        END
        ELSE
        IS_SQL = 'N'
      END
      ELSE
      IS_SQL = 'N'
     'EXECIO 1'                                                         ,
         'DISKW' FN _SQLGEN_ FM Ø F '(ST' RIGHT(EXTRA_REC, 4, 'Ø')
     'EXECIO' EXTRA_REC DISKR FN FT FM REC_NO_START
     'EXECIO' EXTRA_REC DISKW FN _SQLGEN_ FM Ø F
    END
    REC_NO_START = REC_NO_SQL
  END
END
```

PREPT EXEC

```
/*********************************************************************/
/***                                                    ***         ***/
/*** PREPT  prepare & translate PL/I, C, COBOL source ***  DG'97  ***/
/***                                                    ***         ***/
/*********************************************************************/
/***   SIZE ØØØ55  VER 1.Ø MOD ØØ  TIME 16:4Ø:Ø9  DATE 2Ø/Ø6/97   ***/
/*********************************************************************/

  ARG CALL_SQL LANG FN LANG_OPTIONS
  I = ((INDEX('PLICOBC  ', LEFT(LANG, 3))) - 1) / 3 + 1
  IF I < 1 ! I > 3 THEN
  DO
    SAY '--- Wrong host language - must be PLI, COB, or C'
    EXIT
  END
  SET CMSTYPE HT
  MAKEBUF
  STATE FN MACRO A
  IF RC ¬= Ø THEN
  DO
    SET CMSTYPE RT
    SAY '---' FN 'MACRO A not found'
    EXIT
  END
  FT.1 = 'PLIOPT'
  FT.2 = 'COBOL'
  FT.3 = 'C'
  SET CMSTYPE RT
  IF CALL_SQL = 'P' THEN
  DO
    SQLPREP LANG PP '(PREP='FN') IN ('FN 'MACRO A)'
    IF RC > 4 THEN
    DO
      SAY '--- Translation cancelled due to errors when preprocessing'
      EXIT
    END
    EXSTRSQL FN FT.I A
    SAY COPIES('/',25)COPIES('\',25)
    SAY CENTER('Generated code by preprocessor was stored', 5Ø)
    SAY COPIES('\',25)COPIES('/',25)
  END
  ELSE
  DO
    SET CMSTYPE HT
    ERASE FN FT.I A
    SET CMSTYPE RT
```

```
  INCLSQL LANG FN MACRO A
  SAY COPIES('/',25)COPIES('\',25)
  SAY CENTER('Generated code by preprocessor was included', 50)
  SAY COPIES('\',25)COPIES('/',25)
END
IF I = 3 THEN
CC FN FT.3 A
ELSE
FT.I FN LANG_OPTIONS
```

GETTING READY

INSTALL EXEC should be used to generate executable code. Do not forget that the first step must be the preprocessing of source code. After this, if SQL/DS statements do not change, you can debug this source without preprocessing. Preprocessing is required only if SQL/DS statements are modified.

Dobrin Goranov
Information Services Co (Bulgaria)

Call for papers

Why not share your expertise and earn money at the same time? *VM Update* is looking for REXX EXECs, macros, program code, etc, that experienced VMers have written to make their life, or the lives of their users, easier. We will publish it (after vetting by our expert panel) and send you a cheque when the article is published. Articles can be of any length and can be sent or e-mailed to Robert Burgess at any of the addresses shown on page 2. Why not call now for a free copy of our *Notes for contributors*?

Dynamic menus system for CMS – part 3

This month we continue the code for the on-line administration utilities that go with the dynamic menus system for CMS.

```
WHEN VALUE(AIDKEY)='ENTER' THEN DO              /* PROCESS ENTER KEY */
     'PIPE (ENDCHAR ?) STEM SCR_OUT.',
     '| A: FANOUT ',
     '| TAKE FIRST | VAR N_MENU',
     '?A: | TAKE LAST | VAR UPDYN ',
     '?A: | DROP 1 | DROP LAST | STEM CMND.'
      IF MENU¬=N_MENU THEN DO
         MENU=N_MENU
         LATTR.='C031410042F4'X;ATTR.='C031410042F4'X
         MENLEXEC.='';MENLDESC.='';MENLTYPE.='';TOPBOT=1
         SIGNAL LOAD_ARRAYS
       END
         IF STRIP(N_MENU)='' THEN DO
        'XMITMSG 50 (APPLID TAF CALLER XLI NOCOMP VAR'
         UPDYN='N'
         SIGNAL XLINES
         END
         DO I=1 TO CMND.0 BY 4
           CMND.I=STRIP(CMND.I)
           SELECT
           WHEN CMND.I='' THEN DO               /* NULL CMND */
            NOP
            END /* END NULL */

           WHEN CMND.I¬='' & UPDYN¬='Y' THEN DO   /* UPDATE NOT YES */
           'XMITMSG 61 (APPLID TAF CALLER XLI NOCOMP VAR'
            SIGNAL XLINES
            END /* END NULL */

           WHEN CMND.I='A' THEN DO              /* ADD LINE */
            M=(I%4)+1;D=I+1;E=I+2;T=I+3
            IF (CMND.0%4) >18 THEN DO
           'XMITMSG 65 MENU (APPLID TAF CALLER XLI NOCOMP VAR'
            UPDYN='N'
            SIGNAL XLINES
            END
      IF POS((STRIP(CMND.T)),'DELETED MENU MENPROC EXEC REXX')=0 THEN DO
           'XMITMSG 66 CMND.T (APPLID TAF CALLER XLI NOCOMP VAR'
            UPDYN='N'
            SIGNAL XLINES
            END
            NEW_MENULINE=LEFT(MENU,9)||LEFT(CMND.E,10),
```

```
        ||LEFT(CMND.D,19)||LEFT(CMND.T,8)
       'PIPE < MENU XLINES A ',
       '| LOCATE (1.9) /'LEFT(MENU,9)'/',
       '| DROP LAST ',
       '| APPEND LITERAL 'NEW_MENULINE,
       '| APPEND LITERAL 'LEFT(MENU,9)||'***END***',
       '| > 'MENU' XLINES A3'
       'PIPE < MENU XLINES A ',
       '| NLOCATE (1.9) /'LEFT(MENU,9)'/',
       '| APPEND < 'MENU XLINES A3,
       '| > MENU XLINES A'
       'XMITMSG 63 MENU (APPLID TAF CALLER XLI NOCOMP VAR'
        END /* END ADD */

      WHEN CMND.I='D' THEN DO              /* DELETE LINE */
       M=(I%4)+1
     'PIPE < MENU XLINES | NLOCATE /'MENULINE.M'/| > MENU XLINES A'
       K=((M*2)+18)
       J=((M*2)+16)
       /* UPDATE AUTHORIZATION FILE */
       'PIPE (ENDCHAR ?) < MENU XAUTH A',
       '| A:NLOCATE (1Ø.8) /'MENU'/',
       '| B:FANINANY',
       '| > MENU XAUTH A',
       '? A:',
       '| LOCATE (1Ø.8) /'MENU'/',
       '| SPECS 1-* 1 'K'-* 'J '| B:'
       'XMITMSG 62 MENU M (APPLID TAF CALLER XLI NOCOMP VAR'
        END /* END DELETE */

      WHEN CMND.I='C' THEN DO              /* CHANGE LINE */
       M=(I%4)+1;D=I+1;E=I+2;T=I+3
       J=((M*2)+16)
  IF POS((STRIP(CMND.T)),'DELETED MENU MENPROC EXEC REXX')=Ø THEN DO
       'XMITMSG 66 CMND.T (APPLID TAF CALLER XLI NOCOMP VAR'
        UPDYN='N'
        SIGNAL XLINES
        END
        NEW_MENULINE=LEFT(MENU,9)||LEFT(CMND.E,1Ø),
        ||LEFT(CMND.D,19)||LEFT(CMND.T,8)
       'PIPE < MENU XLINES | CHANGE /'MENULINE.M'/'NEW_MENULINE'/',
       '| > MENU XLINES A'
        /* UPDATE AUTHORIZATION FILE */
       'PIPE (ENDCHAR ?) < MENU XAUTH A',
       '| A:NLOCATE (1Ø.8) /'MENU'/',
       '| B:FANINANY',
       '| > MENU XAUTH A',
       '? A:',
       '| LOCATE (1Ø.8) /'MENU'/',
       '| SPECS 1-* 1 /! / 'J'| B:'
```

```
          'XMITMSG 64 MENU M (APPLID TAF CALLER XLI NOCOMP VAR'
           END /* END CHANGE */

          OTHERWISE DO
         'XMITMSG 6Ø CMND.I (APPLID TAF CALLER XLI NOCOMP VAR'
          UPDYN='N'
           END /* END OTHERWISE */

          END /* END SELECT COMMAND */
         END /* END DO CMND. */
      UPDYN='N'
      LATTR.='CØ31410042F4'X;ATTR.='CØ31410042F4'X
      MENLEXEC.='';MENLDESC.='';MENLTYPE.='';TOPBOT=1;MENULINE.=''
      SIGNAL LOAD_ARRAYS
     END /* END ENTER */
WHEN VALUE(AIDKEY)='PF11' THEN DO
                                /* SCROLL FORWARD THROUGH MENU LIST */
     DROP_MENU=DROP_MENU+1
     IF DROP_MENU >= MENUS.Ø THEN DROP_MENU=DROP_MENU-MENUS.Ø
     'PIPE STEM MENUS.',
     '| DROP 'DROP_MENU,
     '| TAKE 1 ',
     '| VAR MENU'
     LATTR.='CØ31410042F4'X;ATTR.='CØ31410042F4'X
     MENLEXEC.='';MENLDESC.='';MENLTYPE.='';TOPBOT=1
     SIGNAL LOAD_ARRAYS
     END /* END PF11 */
WHEN VALUE(AIDKEY)='PF1Ø' THEN DO
                               /* SCROLL BACKWARD THROUGH MENU LIST */
     DROP_MENU=DROP_MENU-1
     IF DROP_MENU < Ø THEN DROP_MENU=MENUS.Ø-1
     'PIPE STEM MENUS.',
     '| DROP 'DROP_MENU,
     '| TAKE 1 ',
     '| VAR MENU'
     LATTR.='CØ31410042F4'X;ATTR.='CØ31410042F4'X
     MENLEXEC.='';MENLDESC.='';MENLTYPE.='';TOPBOT=1
     SIGNAL LOAD_ARRAYS
     END /* END PF1Ø */
WHEN VALUE(AIDKEY)='PFØ6' THEN DO               /* DELETE WHOLE MENU */
     IF STRIP(SCR_OUT.1)='' THEN DO
     'XMITMSG 71 (APPLID TAF CALLER XLI NOCOMP VAR'
     UPDYN='N'
     SIGNAL XLINES
      END
       L=SCR_OUT.Ø;UPDYN=SCR_OUT.L
       IF UPDYN='Y' THEN DO
        'PIPE < MENU XLINES A',
        '| NLOCATE (1.8) /'SCR_OUT.1'/',
        '| > MENU XLINES A'
```

```
        'XMITMSG 72 SCR_OUT.1 (APPLID TAF CALLER XLI NOCOMP VAR'
        SIGNAL RESET_ALL
        END
          ELSE DO
         'XMITMSG 61 (APPLID TAF CALLER XLI NOCOMP VAR'
          SIGNAL XLINES
         END
     END /* END PFØ6 */
WHEN VALUE(AIDKEY)='PFØ7' THEN DO                    /* TOP OF MENU */
     TOPBOT=1
     LATTR.='CØ31410042F4'X;ATTR.='CØ31410042F4'X
     MENLEXEC.='';MENLDESC.='';MENLTYPE.=''
     SIGNAL LOAD_ARRAYS
     END /* END PFØ7 */
WHEN VALUE(AIDKEY)='PFØ8' THEN DO                   /* BOTTOM OF MENU */
     TOPBOT=MENLEXEC.Ø-9
     IF TOPBOT<=Ø THEN TOPBOT=1
     LATTR.='CØ31410042F4'X;ATTR.='CØ31410042F4'X
     MENLEXEC.='';MENLDESC.='';MENLTYPE.=''
     SIGNAL LOAD_ARRAYS
     END /* END PFØ8 */
WHEN VALUE(AIDKEY)='PFØ9' THEN DO                    /* CALL ADD MENU */
     IF STRIP(SCR_OUT.1)='' THEN DO
     'XMITMSG 71 (APPLID TAF CALLER XLI NOCOMP VAR'
     UPDYN='N'
     SIGNAL XLINES
      END
     CALL ZLINES SCR_OUT.1
     SIGNAL RESET_ALL
     END /* END PFØ9 */
WHEN VALUE(AIDKEY)='PF13' THEN DO                          /* TEST */
     SAY 'MENU' MENU
     SAY 'DATA OUT 1' SCR_OUT.1
     SAY 'UPDYN' UPDYN
     SIGNAL XLINES
     RETURN
     END /* END PF13 TEST */
WHEN VALUE(AIDKEY)='PF24' | VALUE(AIDKEY)='PF12' THEN EXIT
WHEN VALUE(AIDKEY)='CLEAR' | VALUE(AIDKEY)='PA2' THEN SIGNAL RESET_ALL
OTHERWISE DO
     'XMITMSG 1 'VALUE(AIDKEY)' (APPLID TAF CALLER XLI NOCOMP VAR'
     SIGNAL XLINES
     END /* END OTHERWISE */
END /* END SELECT */
RETURN
/*
*/
```

XLINES.MEN

```
OPTION===EXEC======DESCRIPTION========TYPE========
0000000001111111111222222222233333333334444444445
1234567890123456789012345678901234567890123456789Ø
OPTION===EXEC======DESCRIPTION========TYPE========
OAMENU   KOTERET   ONLINE ADMIN MENU  TITLE
OAMENU   XAUTH     AUTHORIZATIONS     EXEC
OAMENU   XLINES    MENUS              EXEC
OAMENU   XSTATMS   PROCEDURES         EXEC
OAMENU   XPASW     PASSWORDS DISPLAY  EXEC
OAMENU   GOOUT     EXIT               EXEC
OAMENU   ***END***
FOCUS-P  KOTERET   FOCUS REPS-PROD    TITLE
FOCUS-P  REPORTS   REPS - REPORTS     MENPROC
FOCUS-P  EPICP     REPS - EPIC        MENPROC
FOCUS-P  SASP      REPS - SAS         MENPROC
FOCUS-P  ACCNTP    REPS - ACCNT       MENPROC
FOCUS-P  ***END***
FOCUS-1  KOTERET   FOCUS REPS-TEST    TITLE
FOCUS-1  REPSEX    REPS - REPORTS     MENPROC
FOCUS-1  EPICE     REPS - EPIC        MENPROC
FOCUS-1  SASE      REPS - SAS         MENPROC
FOCUS-1  ACCNT     REPS - ACCNT       MENPROC
FOCUS-1  TESTE     REPS - TEST        MENPROC
FOCUS-1  ***END***
FOCUS-2  KOTERET   FOCUS APPLS        TITLE
FOCUS-2  TVIOTREP  TVIOT REPORTS      EXEC
FOCUS-2  FORM3     FORM3 REPORTS      MENPROC
FOCUS-2  SHIVKH    HAIM SHIVUK        EXEC
FOCUS-2  DICTI     DICTIONARY         MENPROC
FOCUS-2  MAFSUPER  SUPER-MAFIL        MENPROC
FOCUS-2  SUPER     SUPER-USER         MENPROC
FOCUS-2  DELETALL  HOUSEKEEPING       EXEC
FOCUS-2  ***END***
SYSCENT  KOTERET   OPERATIONS MENU    TITLE
SYSCENT  TLMENU    TEMPUS-LINK        MENU
SYSCENT  OPER      VM-OPERATOR        EXEC
SYSCENT  VMBACKUP  VM-BACKUP          EXEC
SYSCENT  VMBATCH   VM-BATCH           EXEC
SYSCENT  EXPLORE   EXPLORE-VM         EXEC
SYSCENT  FOCUS     FOCUS              DELETED
SYSCENT  INTM      CMS SUBSET         EXEC
SYSCENT  GOOUT     EXIT               EXEC
SYSCENT  ***END***
CMSUTIL  KOTERET   CMS UTILITIES      TITLE
CMSUTIL  QPWR      VIEW PNET QUEUE    EXEC
CMSUTIL  AVIEREP   RUN EREP           EXEC
CMSUTIL  WHERIZIT  FIND MY STRING     EXEC
CMSUTIL  DOAR      E-MAIL             EXEC
```

```
CMSUTIL  GOOUT     EXIT               EXEC
CMSUTIL  ***END***
TLMENU   KOTERET    TEMPUS-LINK MENU  TITLE
TLMENU   TEMPUSPC  ELEMENTAR          EXEC
TLMENU   TEMPOR    GVIA ELEMENTAR     EXEC
TLMENU   TLINIT    GENERATE LIBRARY   EXEC
TLMENU   TLPRDX    TRANSFER TO PRDX   EXEC
TLMENU   TEMPUSGV  GVIA ELMN GENTIUM  EXEC
TLMENU   TEMPSAS   TRANSFER TO SAS    EXEC
TLMENU   ***END***
MLTPRT   KOTERET   MULTIPRINT MENU    TITLE
MLTPRT   YPRT12    ACTIVE PRINTERS    EXEC
MLTPRT   YPRT11    DISPLAY PRINTERS   EXEC
MLTPRT   YPRT3     DISPLAY REPORTS    EXEC
MLTPRT   YPRT4     CONSOLE            EXEC
MLTPRT   MPROPER   MP-OPERATOR        EXEC
MLTPRT   GOOUT     EXIT               EXEC
MLTPRT   ***END***
```

XPASW EXEC

```
/* XPASW EXEC */
/* AIDKEY EQUATES */
$F1='PFØ1';$F2='PFØ2';$F3='PFØ3';$F4='PFØ4';$F5='PFØ5';$F6='PFØ6'
$F7='PFØ7';$F8='PFØ8';$F9='PFØ9';$7A='PF1Ø';$7B='PF11';$7C='PF12'
$C1='PF13';$C2='PF14';$C3='PF15';$C4='PF16';$C5='PF17';$C6='PF18'
$C7='PF19';$C8='PF2Ø';$C9='PF21';$4A='PF22';$4B='PF23';$4C='PF24'
$Ø1='PA1';$6E='PA2';$7D='ENTER';$6D='CLEAR'
'SET LANGUAGE (ADD TAF USER'  /* MESSAGE REPOSITORY IS TAFUME REPOS */
'VMFCLEAR'
'PIPE < LOG1 XPASW | DROP FIRST | COUNT LINES | VAR UIDS'
RESET_ALL:
MESSAGE.1=COPIES(' ',56)
UID.='';PSWD.='';DAT.='';PPSWD.='';PDAT.=''
DROP=Ø

LOAD_ARRAYS:
'PIPE (ENDCHAR ?) < LOG1 XPASW ',
'| DROP FIRST ',
'| SORT 1.8',
'| DROP 'DROP,
'| A: FANOUT ',
'| SPECS 1.8 1 | STEM UID.',
'?A:| SPECS 1Ø.8 1 | STEM PSWD.',
'?A:| SPECS /X/ 1 19.5 2 /F/ 7 | XPASW | STEM DAT.',
'?A:| SPECS 39.8 1 | STEM PPSWD.',
'?A:| SPECS /X/ 1 48.5 2 /F/ 7 | XPASW | STEM PDAT.'

 XPASW :
```

```
 SCREEN='8003'X||,
'1100002903C020410042F3'X||COPIES('=',77)||'11004E1DF0'X||,
'11004F2903C020410042F3'X||'|'||'1100511DF0'X||,
'1100632903C02041F242F5'X||' DYNAMIC MENUS ADMINISTRATION UTILITIES
'||'11008C1DF0'X||,
'11009D2903C020410042F3'X||'|'||'11009F1DF0'X||,
'11009F2903C020410042F3'X||'|'||COPIES('=',77)||'|'||'1100EF1DF0'X||,
'1100EF2903C020410042F3'X||'|'||'1100F11DF0'X||,
'1101092903C02041F242F5'X||' ACCESS PASSWORDS DISPLAY
'||'1101251DF0'X||,
'11013D2903C020410042F3'X||'|'||'11013F1DF0'X||,
'11013F2903C020410042F3'X||'|'||COPIES('=',77)||'|'||'11018F1DF0'X||,
'11018F2903C020410042F3'X||'|'||'1101911DF0'X||,
'1101932903C02041F242F7'X||'  USERID  '||'11019E1DF0'X||,
'11019F2903C02041F242F7'X||' PASSWORD '||'1101AA1DF0'X||,
'1101AB2903C02041F242F7'X||' UPDATED ON '||'1101B81DF0'X||,
'1101B92903C02041F242F7'X||' PREVIOUS '||'1101C41DF0'X||,
'1101C52903C02041F242F7'X||' UPDATED ON '||'1101D21DF0'X||,
'1101DD2903C020410042F3'X||'| |'||'1101E21DF0'X||,
'11022D2903C020410042F3'X||'|'||'11022F1DF0'X||,
'11022F2903C020410042F3'X||'|'||'1102311DF0'X||,
'1102342903C020410042F4'X||UID.1   ||'11023D1DF0'X||,
'1102402903C020410042F4'X||PSWD.1  ||'1102491DF0'X||,
'11024C2903C020410042F4'X||DAT.1   ||'1102571DF0'X||,
'11025A2903C020410042F4'X||PPSWD.1 ||'1102631DF0'X||,
'1102662903C020410042F4'X||PDAT.1  ||'1102711DF0'X||,
'11027D2903C020410042F3'X||'|'||'11027F1DF0'X||,
'11027F2903C020410042F3'X||'|'||'1102811DF0'X||,
'1102842903C020410042F4'X||UID.2   ||'11028D1DF0'X||,
'1102902903C020410042F4'X||PSWD.2  ||'1102991DF0'X||,
'11029C2903C020410042F4'X||DAT.2   ||'1102A71DF0'X||,
'1102AA2903C020410042F4'X||PPSWD.2 ||'1102B31DF0'X||,
'1102B62903C020410042F4'X||PDAT.2  ||'1102C11DF0'X||,
'1102CD2903C020410042F3'X||'|'||'1102CF1DF0'X||,
'1102CF2903C020410042F3'X||'|'||'1102D11DF0'X||,
'1102D42903C020410042F4'X||UID.3   ||'1102DD1DF0'X||,
'1102E02903C020410042F4'X||PSWD.3  ||'1102E91DF0'X||,
'1102EC2903C020410042F4'X||DAT.3   ||'1102F71DF0'X||,
'1102FA2903C020410042F4'X||PPSWD.3 ||'1103031DF0'X||,
'1103062903C020410042F4'X||PDAT.3  ||'1103111DF0'X||,
'11031D2903C020410042F3'X||'|'||'11031F1DF0'X||,
'11031F2903C020410042F3'X||'|'||'1103211DF0'X||,
'1103242903C020410042F4'X||UID.4   ||'11032D1DF0'X||,
'1103302903C020410042F4'X||PSWD.4  ||'1103391DF0'X||,
'11033C2903C020410042F4'X||DAT.4   ||'1103471DF0'X||,
'11034A2903C020410042F4'X||PPSWD.4 ||'1103531DF0'X||,
'1103562903C020410042F4'X||PDAT.4  ||'1103611DF0'X||,
'11036D2903C020410042F3'X||'|'||'11036F1DF0'X||,
'11036F2903C020410042F3'X||'|'||'1103711DF0'X||,
'1103742903C020410042F4'X||UID.5   ||'11037D1DF0'X||,
```

```
'1103802903C020410042F4'X||PSWD.5  ||'1103891DF0'X||,
'11038C2903C020410042F4'X||DAT.5   ||'1103971DF0'X||,
'11039A2903C020410042F4'X||PPSWD.5 ||'1103A31DF0'X||,
'1103A62903C020410042F4'X||PDAT.5  ||'1103B11DF0'X||,
'1103BD2903C020410042F3'X||'|'||'1103BF1DF0'X||,
'1103BF2903C020410042F3'X||'|'||'1103C11DF0'X||,
'1103C42903C020410042F4'X||UID.6   ||'1103CD1DF0'X||,
'1103D02903C020410042F4'X||PSWD.6  ||'1103D91DF0'X||,
'1103DC2903C020410042F4'X||DAT.6   ||'1103E71DF0'X||,
'1103EA2903C020410042F4'X||PPSWD.6 ||'1103F31DF0'X||,
'1103F62903C020410042F4'X||PDAT.6  ||'1104011DF0'X||,
'11040D2903C020410042F3'X||'|'||'11040F1DF0'X||,
'11040F2903C020410042F3'X||'|'||'1104111DF0'X||,
'1104142903C020410042F4'X||UID.7   ||'11041D1DF0'X||,
'1104202903C020410042F4'X||PSWD.7  ||'1104291DF0'X||,
'11042C2903C020410042F4'X||DAT.7   ||'1104371DF0'X||,
'11043A2903C020410042F4'X||PPSWD.7 ||'1104431DF0'X||,
'1104462903C020410042F4'X||PDAT.7  ||'1104511DF0'X||,
'11045D2903C020410042F3'X||'|'||'11045F1DF0'X||,
'11045F2903C020410042F3'X||'|'||'1104611DF0'X||,
'1104642903C020410042F4'X||UID.8   ||'11046D1DF0'X||,
'1104702903C020410042F4'X||PSWD.8  ||'1104791DF0'X||,
'11047C2903C020410042F4'X||DAT.8   ||'1104871DF0'X||,
'11048A2903C020410042F4'X||PPSWD.8 ||'1104931DF0'X||,
'1104962903C020410042F4'X||PDAT.8  ||'1104A11DF0'X||,
'1104AD2903C020410042F3'X||'|'||'1104AF1DF0'X||,
'1104AF2903C020410042F3'X||'|'||'1104B11DF0'X||,
'1104B42903C020410042F4'X||UID.9   ||'1104BD1DF0'X||,
'1104C02903C020410042F4'X||PSWD.9  ||'1104C91DF0'X||,
'1104CC2903C020410042F4'X||DAT.9   ||'1104D71DF0'X||,
'1104DA2903C020410042F4'X||PPSWD.9 ||'1104E31DF0'X||,
'1104E62903C020410042F4'X||PDAT.9  ||'1104F11DF0'X||,
'1104FD2903C020410042F3'X||'|'||'1104FF1DF0'X||,
'1104FF2903C020410042F3'X||'|'||'1105011DF0'X||,
'1105042903C020410042F4'X||UID.10  ||'11050D1DF0'X||,
'1105102903C020410042F4'X||PSWD.10 ||'1105191DF0'X||,
'11051C2903C020410042F4'X||DAT.10  ||'1105271DF0'X||,
'11052A2903C020410042F4'X||PPSWD.10||'1105331DF0'X||,
'1105362903C020410042F4'X||PDAT.10 ||'1105411DF0'X||,
'11054D2903C020410042F3'X||'|'||'11054F1DF0'X||,
'11054F2903C020410042F3'X||'|'||'1105511DF0'X||,
'1105542903C020410042F4'X||UID.11  ||'11055D1DF0'X||,
'1105602903C020410042F4'X||PSWD.11 ||'1105691DF0'X||,
'11056C2903C020410042F4'X||DAT.11  ||'1105771DF0'X||,
'11057A2903C020410042F4'X||PPSWD.11||'1105831DF0'X||,
'1105862903C020410042F4'X||PDAT.11 ||'1105911DF0'X||,
'11059D2903C020410042F3'X||'|'||'11059F1DF0'X||,
'11059F2903C020410042F3'X||'|'||'1105A11DF0'X||,
'1105A42903C020410042F4'X||UID.12  ||'1105AD1DF0'X||,
'1105B02903C020410042F4'X||PSWD.12 ||'1105B91DF0'X||,
```

```
'1105BC2903C020410042F4'X||DAT.12  ||'1105C71DF0'X||,
'1105CA2903C020410042F4'X||PPSWD.12||'1105D31DF0'X||,
'1105D62903C020410042F4'X||PDAT.12 ||'1105E11DF0'X||,
'1105ED2903C020410042F3'X||'|'||'1105EF1DF0'X||,
'1105EF2903C020410042F3'X||'|'||'1105F11DF0'X||,
'1105F42903C020410042F4'X||UID.13  ||'1105FD1DF0'X||,
'1106002903C020410042F4'X||PSWD.13 ||'1106091DF0'X||,
'11060C2903C020410042F4'X||DAT.13  ||'1106171DF0'X||,
'11061A2903C020410042F4'X||PPSWD.13||'1106231DF0'X||,
'1106262903C020410042F4'X||PDAT.13 ||'1106311DF0'X||,
'11063D2903C020410042F3'X||'|'||'11063F1DF0'X||,
'11063F2903C020410042F3'X||'|'||'1106411DF0'X||,
'1106442903C020410042F4'X||UID.14  ||'11064D1DF0'X||,
'1106502903C020410042F4'X||PSWD.14 ||'1106591DF0'X||,
'11065C2903C020410042F4'X||DAT.14  ||'1106671DF0'X||,
'11066A2903C020410042F4'X||PPSWD.14||'1106731DF0'X||,
'1106762903C020410042F4'X||PDAT.14 ||'1106811DF0'X||,
'11068D2903C020410042F3'X||'|'||'11068F1DF0'X||,
'11068F2903C020410042F3'X||'|'||COPIES('_',77)||'|'||'1106DF1DF0'X||,
'1106DF2903C020410042F3'X||'|'||'1106E11DF0'X||,
'1106F32903C03041F142F2'X||MESSAGE.1||'11072C1DF0'X||,
'11072D2903C020410042F3'X||'|'||'11072F1DF0'X||,
'1107302903C020410042F3'X||COPIES('=',77)||'11077E1DF0'X||,
'11019413'X
'PIPE VAR SCREEN | FULLSCREEN ',
'| SPLIT AT ANYOF /'"11"X'/',
'| A: FANOUT',
'| TAKE FIRST',
'| SPECS /$/ 1 1-1 C2X NEXT',
'| VAR AIDKEY'
MESSAGE.1=COPIES(' ',50)
SELECT
WHEN VALUE(AIDKEY)='PF07' THEN DO                  /* SCROLL BACKWARD */
     DROP=DROP-13
     IF DROP<0 THEN DO
       DROP=0
      'XMITMSG 10 (APPLID TAF CALLER XPA NOCOMP VAR'
      END
      UID.='';PSWD.='';DAT.='';PPSWD.='';PDAT.=''
      SIGNAL LOAD_ARRAYS
     END /* END PF07 */
WHEN VALUE(AIDKEY)='PF08' THEN DO                   /* SCROLL FORWARD */
     DROP=DROP+13
     IF DROP>UIDS THEN DO
       DROP=UIDS-14
       IF DROP<0 THEN DROP=0
      'XMITMSG 11 (APPLID TAF CALLER XPA NOCOMP VAR'
      END
      UID.='';PSWD.='';DAT.='';PPSWD.='';PDAT.=''
      SIGNAL LOAD_ARRAYS
```

```
     END /* END PFØ8 */
WHEN VALUE(AIDKEY)='PF13' THEN DO                                  /* TEST */
     SAY MENU
     SIGNAL XPASW
     END /* END PF13 TEST */
WHEN VALUE(AIDKEY)='PF24' | VALUE(AIDKEY)='PF12' THEN EXIT
WHEN VALUE(AIDKEY)='CLEAR' | VALUE(AIDKEY)='PA2' THEN SIGNAL RESET_ALL
WHEN VALUE(AIDKEY)='ENTER' THEN SIGNAL XPASW
OTHERWISE DO
     'XMITMSG 1 'VALUE(AIDKEY)' (APPLID TAF CALLER XPA NOCOMP VAR'
     SIGNAL XPASW
     END /* END OTHERWISE */
END /* END SELECT */
RETURN
/*
*/
```

XPASW LOG

```
=USERID==PASSWORD=DATE=TIME===HISTORY=DATA============================
MAINT   !MAINT   !9621760038!MAINT   !MAINTABI!9621759531!
759518!
MENUADMN!MENUADMN!9624249167!
FOCUSER !FOCUSER !9624249192!
INFOCENT!INFOCENT!9624249430!
TESTER  !TESTER  !9624249446!
TEMPUS  !TEMPUS  !9624249470!
```

XPASW REXX

```
/* XPASW */
SIGNAL ON ERROR
DO N=1 BY 1
    'READTO RECORD'
    IF RECORD='X     F' THEN OUTPUT='          '
                         ELSE DO
    'CALLPIPE (ESCAPE %) STRLITERAL 'RECORD'| SPECS 1-* C2I 1 ',
    '| SPECS 7.2 1 !%/! 3 5.2 4 !%/! 6 1.4 7 |  VAR OUTPUT'
     END
    'OUTPUT' OUTPUT
END
ERROR:
IF RC=12 THEN RC=0
EXIT RC
```

XSTAMS EXEC

```
/* XSTATMS EXEC */
/* AIDKEY EQUATES */
$F1='PFØ1';$F2='PFØ2';$F3='PFØ3';$F4='PFØ4';$F5='PFØ5';$F6='PFØ6'
$F7='PFØ7';$F8='PFØ8';$F9='PFØ9';$7A='PF1Ø';$7B='PF11';$7C='PF12'
$C1='PF13';$C2='PF14';$C3='PF15';$C4='PF16';$C5='PF17';$C6='PF18'
$C7='PF19';$C8='PF2Ø';$C9='PF21';$4A='PF22';$4B='PF23';$4C='PF24'
$Ø1='PA1';$6E='PA2';$7D='ENTER';$6D='CLEAR'
'SET LANGUAGE (ADD TAF USER'  /* MESSAGE REPOSITORY IS TAFUME REPOS */
LINK_MODES='R RR W WR M MR MW'
'VMFCLEAR'
MESSAGE.1=COPIES(' ',56)

RESET_ALL:
P=Ø
PROCS=''
PROC='        '
EXECNAME='        '
TDSK=' Y '
UID.='        ';O_ADDR.='    ';T_ADDR.='    '
MODE.='  ';LPSWD.='        ';A_MODE.='    ';UPDYN='N'

'PIPE < MENU XSTATMS ',
'| DROP 14',
'| SPECS 1.8 1',
'| SORT UNIQUE',
'| STEM PROCS.'
DO I=1 TO PROCS.Ø
PROCS=PROCS PROCS.I
END

LOAD_PROC_NAME:
IF P=Ø THEN SIGNAL XSTATMS
PROC=PROCS.P

LOAD_ARRAYS:
'PIPE (ENDCHAR ?) < MENU XSTATMS ',
'| DROP 14',
'| LOCATE (1.8) /'PROC'/',
'| DROP FIRST',
'| A: FANOUT | LOCATE (9.4) / EX / | SPECS W3 1 | VAR EXECNAME',
'?A:| LOCATE (9.7) / TDISK / | SPECS W3 1 | VAR TDSK',
'?A:| LOCATE (9.8) / ACCESS / | SPECS W4 1 | STEM A_MODE.',
'?A:| LOCATE (9.6) / LINK / | B: FANOUT | SPECS W3 1 | STEM UID.',
'?B:| SPECS W4 1 | STEM O_ADDR.',
'?B:| SPECS W5 1 | STEM T_ADDR.',
'?B:| SPECS W6 1 | STEM MODE.',
'?B:| SPECS W7 X2C 1 | REVERSE | STEM LPSWD.'
 XSTATMS :
```

```
 SCREEN='8003'X||,
'1100002903C020410042F3'X||COPIES('=',77)||'11004E1DF0'X||,
'11004F2903C020410042F3'X||'|'||'1100511DF0'X||,
'1100632903C02041F242F5'X||' DYNAMIC MENUS ADMINISTRATION UTILITIES
'||'11008C1DF0'X||,
'11009D2903C020410042F3'X||'|'||'11009F1DF0'X||,
'11009F2903C020410042F3'X||'|'||COPIES('=',77)||'|'||'1100EF1DF0'X||,
'1100EF2903C020410042F3'X||'|'||'1100F11DF0'X||,
'1101082903C02041F242F5'X||' PROCEDURES MANAGEMENT UTILITY
'||'1101281DF0'X||,
'11013D2903C020410042F3'X||'|'||'11013F1DF0'X||,
'11013F2903C020410042F3'X||'|'||COPIES('=',77)||'|'||'11018F1DF0'X||,
'11018F2903C020410042F3'X||'|'||'1101911DF0'X||,
'1101942903C02041F242F7'X||' PROC NAME : '||'1101A21DF0'X||,
'1101A52903C00141F442F7'X||PROC||'1101AE1DF0'X||,
'1101B02903C02041F242F7'X||' EXECUTABLE PROGRAM : '||'1101C71DF0'X||,
'1101CA2903C00141F442F7'X||EXECNAME||'1101D31DF0'X||,
'1101DD2903C020410042F3'X||'| |'||'1101E21DF0'X||,
'1101F52903C02041F242F6'X||'PF10/11'||'1101FD1DF0'X||,
'1102002903C02041F242F7'X||' TDISK (Y/N/CYLNUM) : '||'1102171DF0'X||,
'11021A2903C00141F442F7'X||TDSK||'11021F1DF0'X||,
'11022D2903C020410042F3'X||'|'||'11022F1DF0'X||,
'11022F2903C020410042F3'X||'|'||COPIES('_',77)||'|'||'11027F1DF0'X||,
'11027F2903C020410042F3'X||'|'||'1102811DF0'X||,
'1102852903C02041F242F7'X||'LINE'||'11028A1DF0'X||,
'11028E2903C02041F242F7'X||' MDISK  '||'1102971DF0'X||,
'1102992903C02041F242F7'X||'ORIGIN'||'1102A01DF0'X||,
'1102A22903C02041F242F7'X||'TARGET'||'1102A91DF0'X||,
'1102AB2903C02041F242F7'X||'LINK'||'1102B01DF0'X||,
'1102B22903C02041F242F7'X||'  LINK  '||'1102BB1DF0'X||,
'1102BD2903C02041F242F7'X||'ACCESS'||'1102C41DF0'X||,
'1102CD2903C020410042F3'X||'|'||'1102CF1DF0'X||,
'1102CF2903C020410042F3'X||'|'||'1102D11DF0'X||,
'1102D52903C02041F242F7'X||' #  '||'1102DA1DF0'X||,
'1102DE2903C02041F242F7'X||' OWNER  '||'1102E71DF0'X||,
'1102E92903C02041F242F7'X||'V-ADDR'||'1102F01DF0'X||,
'1102F22903C02041F242F7'X||'V-ADDR'||'1102F91DF0'X||,
'1102FB2903C02041F242F7'X||'MODE'||'1103001DF0'X||,
'1103022903C02041F242F7'X||'PASSWORD'||'11030B1DF0'X||,
'11030D2903C02041F242F7'X||' MODE '||'1103141DF0'X||,
'11031D2903C020410042F3'X||'|'||'11031F1DF0'X||,
'11031F2903C020410042F3'X||'|'||'1103211DF0'X||,
'1103262903C030410042F4'X||'1'||'1103281DF0'X||,
'11032E2903C00141F442F4'X||UID.1    ||'1103371DF0'X||,
'11033A2903C00141F442F4'X||O_ADDR.1 ||'11033F1DF0'X||,
'1103432903C00141F442F4'X||T_ADDR.1 ||'1103481DF0'X||,
'11034C2903C00141F442F4'X||MODE.1   ||'11034F1DF0'X||,
'1103522903C00141F442F4'X||LPSWD.1  ||'11035B1DF0'X||,
'11035E2903C00141F442F4'X||A_MODE.1 ||'1103631DF0'X||,
'11036D2903C020410042F3'X||'|'||'11036F1DF0'X||,
```

```
'11036F2903C020410042F3'X||'|'||'1103711DF0'X||,
'1103762903C030410042F4'X||'2'||'1103781DF0'X||,
'11037E2903C00141F442F4'X||UID.2    ||'1103871DF0'X||,
'11038A2903C00141F442F4'X||O_ADDR.2 ||'11038F1DF0'X||,
'1103932903C00141F442F4'X||T_ADDR.2 ||'1103981DF0'X||,
'11039C2903C00141F442F4'X||MODE.2   ||'11039F1DF0'X||,
'1103A22903C00141F442F4'X||LPSWD.2  ||'1103AB1DF0'X||,
'1103AE2903C00141F442F4'X||A_MODE.2 ||'1103B31DF0'X||,
'1103BD2903C020410042F3'X||'|'||'1103BF1DF0'X||,
'1103BF2903C020410042F3'X||'|'||'1103C11DF0'X||,
'1103C62903C030410042F4'X||'3'||'1103C81DF0'X||,
'1103CE2903C00141F442F4'X||UID.3    ||'1103D71DF0'X||,
'1103DA2903C00141F442F4'X||O_ADDR.3 ||'1103DF1DF0'X||,
'1103E32903C00141F442F4'X||T_ADDR.3 ||'1103E81DF0'X||,
'1103EC2903C00141F442F4'X||MODE.3   ||'1103EF1DF0'X||,
'1103F22903C00141F442F4'X||LPSWD.3  ||'1103FB1DF0'X||,
'1103FE2903C00141F442F4'X||A_MODE.3 ||'1104031DF0'X||,
'11040D2903C020410042F3'X||'|'||'11040F1DF0'X||,
'11040F2903C020410042F3'X||'|'||'1104111DF0'X||,
'1104162903C030410042F4'X||'4'||'1104181DF0'X||,
'11041E2903C00141F442F4'X||UID.4    ||'1104271DF0'X||,
'11042A2903C00141F442F4'X||O_ADDR.4 ||'11042F1DF0'X||,
'1104332903C00141F442F4'X||T_ADDR.4 ||'1104381DF0'X||,
'11043C2903C00141F442F4'X||MODE.4   ||'11043F1DF0'X||,
'1104422903C00141F442F4'X||LPSWD.4  ||'11044B1DF0'X||,
'11044E2903C00141F442F4'X||A_MODE.4 ||'1104531DF0'X||,
'11045D2903C020410042F3'X||'|'||'11045F1DF0'X||,
'11045F2903C020410042F3'X||'|'||'1104611DF0'X||,
'1104662903C030410042F4'X||'5'||'1104681DF0'X||,
'11046E2903C00141F442F4'X||UID.5    ||'1104771DF0'X||,
'11047A2903C00141F442F4'X||O_ADDR.5 ||'11047F1DF0'X||,
'1104832903C00141F442F4'X||T_ADDR.5 ||'1104881DF0'X||,
'11048C2903C00141F442F4'X||MODE.5   ||'11048F1DF0'X||,
'1104922903C00141F442F4'X||LPSWD.5  ||'11049B1DF0'X||,
'11049E2903C00141F442F4'X||A_MODE.5 ||'1104A31DF0'X||,
'1104AD2903C020410042F3'X||'|'||'1104AF1DF0'X||,
'1104AF2903C020410042F3'X||'|'||'1104B11DF0'X||,
'1104B62903C030410042F4'X||'6'||'1104B81DF0'X||,
'1104BE2903C00141F442F4'X||UID.6    ||'1104C71DF0'X||,
'1104CA2903C00141F442F4'X||O_ADDR.6 ||'1104CF1DF0'X||,
'1104D32903C00141F442F4'X||T_ADDR.6 ||'1104D81DF0'X||,
'1104DC2903C00141F442F4'X||MODE.6   ||'1104DF1DF0'X||,
'1104E22903C00141F442F4'X||LPSWD.6  ||'1104EB1DF0'X||,
'1104EE2903C00141F442F4'X||A_MODE.6 ||'1104F31DF0'X||,
'1104FD2903C020410042F3'X||'|'||'1104FF1DF0'X||,
'1104FF2903C020410042F3'X||'|'||'1105011DF0'X||,
'1105062903C030410042F4'X||'7'||'1105081DF0'X||,
'11050E2903C00141F442F4'X||UID.7    ||'1105171DF0'X||,
'11051A2903C00141F442F4'X||O_ADDR.7 ||'11051F1DF0'X||,
'1105232903C00141F442F4'X||T_ADDR.7 ||'1105281DF0'X||,
```

```
'11052C2903C00141F442F4'X||MODE.7   ||'11052F1DF0'X||,
'1105322903C00141F442F4'X||LPSWD.7  ||'11053B1DF0'X||,
'11053E2903C00141F442F4'X||A_MODE.7 ||'1105431DF0'X||,
'11054D2903C020410042F3'X||'|'||'11054F1DF0'X||,
'11054F2903C020410042F3'X||'|'||'1105511DF0'X||,
'1105562903C030410042F4'X||'8'||'1105581DF0'X||,
'11055E2903C00141F442F4'X||UID.8    ||'1105671DF0'X||,
'11056A2903C00141F442F4'X||O_ADDR.8 ||'11056F1DF0'X||,
'1105732903C00141F442F4'X||T_ADDR.8 ||'1105781DF0'X||,
'11057C2903C00141F442F4'X||MODE.8   ||'11057F1DF0'X||,
'1105822903C00141F442F4'X||LPSWD.8  ||'11058B1DF0'X||,
'11058E2903C00141F442F4'X||A_MODE.8 ||'1105931DF0'X||,
'11059D2903C020410042F3'X||'|'||'11059F1DF0'X||,
'11059F2903C020410042F3'X||'|'||'1105A11DF0'X||,
'1105A62903C030410042F4'X||'9'||'1105A81DF0'X||,
'1105AE2903C00141F442F4'X||UID.9    ||'1105B71DF0'X||,
'1105BA2903C00141F442F4'X||O_ADDR.9 ||'1105BF1DF0'X||,
'1105C32903C00141F442F4'X||T_ADDR.9 ||'1105C81DF0'X||,
'1105CC2903C00141F442F4'X||MODE.9   ||'1105CF1DF0'X||,
'1105D22903C00141F442F4'X||LPSWD.9  ||'1105DB1DF0'X||,
'1105DE2903C00141F442F4'X||A_MODE.9 ||'1105E31DF0'X||,
'1105ED2903C020410042F3'X||'|'||'1105EF1DF0'X||,
'1105EF2903C020410042F3'X||'|'||'1105F11DF0'X||,
'1105F52903C030410042F4'X||'10'||'1105F81DF0'X||,
'1105FE2903C00141F442F4'X||UID.10   ||'1106071DF0'X||,
'11060A2903C00141F442F4'X||O_ADDR.10||'11060F1DF0'X||,
'1106132903C00141F442F4'X||T_ADDR.10||'1106181DF0'X||,
'11061C2903C00141F442F4'X||MODE.10  ||'11061F1DF0'X||,
'1106222903C00141F442F4'X||LPSWD.10 ||'11062B1DF0'X||,
'11062E2903C00141F442F4'X||A_MODE.10||'1106331DF0'X||,
'11063D2903C020410042F3'X||'|'||'11063F1DF0'X||,
'11063F2903C020410042F3'X||'|'||'1106411DF0'X||,
'1106452903C030410042F4'X||'11'||'1106481DF0'X||,
'11064E2903C00141F442F4'X||UID.11   ||'1106571DF0'X||,
'11065A2903C00141F442F4'X||O_ADDR.11||'11065F1DF0'X||,
'1106632903C00141F442F4'X||T_ADDR.11||'1106681DF0'X||,
'11066C2903C00141F442F4'X||MODE.11  ||'11066F1DF0'X||,
'1106722903C00141F442F4'X||LPSWD.11 ||'11067B1DF0'X||,
'11067E2903C00141F442F4'X||A_MODE.11||'1106831DF0'X||,
'11068D2903C020410042F3'X||'|'||'11068F1DF0'X||,
'11068F2903C020410042F3'X||'|'||COPIES('_',77)||'|'||'1106DF1DF0'X||,
'1106DF2903C020410042F3'X||'|'||'1106E11DF0'X||,
'1106E42903C02041F242F6'X||'UPDATE(Y/N)?'||'1106F11DF0'X||,
'1106F12903C00141F442F6'X||UPDYN||'1105F31DF0'X||,
'1106F32903C03041F142F2'X||MESSAGE.1||'11072C1DF0'X||,
'11072D2903C020410042F3'X||'|'||'11072F1DF0'X||,
'1107302903C020410042F3'X||COPIES('=',77)||'11077E1DF0'X||,
'1101A613'X
'PIPE (ENDCHAR ?) VAR SCREEN | FULLSCREEN ',
'| SPLIT AT ANYOF /'"11"X'/',
```

```
'| A: FANOUT',
'| TAKE FIRST',
'| SPECS /$/ 1 1-1 C2X NEXT',
'| VAR AIDKEY',
'? A:',
'| DROP FIRST',
'| SPECS 3-* 1',
'| STEM SCR_OUT.'
MESSAGE.1=COPIES(' ',5Ø)
SELECT
WHEN VALUE(AIDKEY)='ENTER' THEN DO              /* PROCESS ENTER KEY */
    'PIPE (ENDCHAR ?) STEM SCR_OUT. | A: FANOUT',
       '| TAKE FIRST | VAR N_PROC',
    '?A:| DROP 1 | TAKE 1 | VAR EXECNAME ',
    '?A:| DROP 2 | TAKE 1 | VAR TDSK ',
    '?A:| TAKE LAST | VAR UPDYN ',
    '?A:| DROP 3 | DROP LAST | JOIN 5 X72 | B:FANOUT',
       '| SPECS WS 72 W1 1 | STEM UID.',
    '?B:| SPECS WS 72 W2 STRIP 1 | STEM O_ADDR.',
    '?B:| SPECS WS 72 W3 STRIP 1 | STEM T_ADDR.',
    '?B:| SPECS WS 72 W4 1 | STEM MODE.',
    '?B:| SPECS WS 72 W5 STRIP 1 | STEM LPSWD.',
    '?B:| SPECS WS 72 W6 1 | STEM A_MODE.'
  IF STRIP(N_PROC)='' THEN DO                       /* PROCNAME NULL   */
  'XMITMSG 8Ø (APPLID TAF CALLER XST NOCOMP VAR'
   UPDYN='N';P=Ø
   SIGNAL XSTATMS
    END /* END PROCNAME NULL */
  IF UPDYN¬='Y' THEN DO                             /* UPDATE FLAG OFF */
    IF STRIP(N_PROC)¬=STRIP(PROC) THEN DO           /* CHANGE PROCNAME */
     IF FIND(PROCS,N_PROC)¬=Ø THEN DO               /* EXISTING PROC   */
      PROC=N_PROC
      UID.='        ';O_ADDR.='    ';T_ADDR.='    '
      MODE.='  ';LPSWD.='        ';A_MODE.='    ';UPDYN='N'
      P=Ø
      SIGNAL LOAD_ARRAYS
      END /* END EXISTING PROC */
                           ELSE DO          /* PROC DOES NOT EXIST */
     'XMITMSG 81 'N_PROC' (APPLID TAF CALLER XST NOCOMP VAR'
      P=Ø;PROC=N_PROC
      UID.='        ';O_ADDR.='    ';T_ADDR.='    '
      MODE.='  ';LPSWD.='        ';A_MODE.='    ';UPDYN='N'
      SIGNAL XSTATMS
      END /* END NON-EXISTING PROC */
     END /* END NEW PROCNAME */
    END /* END UPDYN=N */
              ELSE DO                               /* UPDATE FLAG ON  */
    IF STRIP(EXECNAME)='' THEN DO                   /* EXECNAME NULL */
   'XMITMSG 82 (APPLID TAF CALLER XST NOCOMP VAR'
    UPDYN='N';P=Ø;PROC=N_PROC
```

```
SIGNAL XSTATMS
 END /* END EXECNAME NULL */
M=Ø                                          /* ZERO LINES COUNTER */
DO N=1 TO UID.Ø                              /* CHECK LINK PHRASES */
 IF STRIP(UID.N)='' THEN ITERATE
 IF STRIP(O_ADDR.N)='' THEN DO
'XMITMSG 83 (APPLID TAF CALLER XST NOCOMP VAR'
 UPDYN='N';P=Ø;PROC=N_PROC
 SIGNAL XSTATMS
  END
 IF STRIP(T_ADDR.N)='' THEN T_ADDR.N=O_ADDR.N
 IF FIND(LINK_MODES,STRIP(MODE.N))=Ø THEN DO
'XMITMSG 84 (APPLID TAF CALLER XST NOCOMP VAR'
 UPDYN='N';P=Ø;PROC=N_PROC
 SIGNAL XSTATMS
  END
 IF STRIP(A_MODE.N)='' THEN DO
'XMITMSG 85 (APPLID TAF CALLER XST NOCOMP VAR'
 UPDYN='N';P=Ø;PROC=N_PROC
 SIGNAL XSTATMS
  END
```

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

Yaakov J Hazan
Technical Support Manager
Ynon Technologies & Computers Ltd (Israel)

We'd like to hear from anyone involved in beta testing the latest e-business hardware and software from IBM (announced in October). We'd like to hear about your experiences and whether you'd recommend the new products to other VM users – and if not, why not. Articles can be of any length and can be sent or e-mailed to Robert Burgess at any of the addresses shown on page 2. Why not call now for a free copy of our *Notes for contributors*?