131 VM

July 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.

Multiplatform command scheduler

CLKQUEUE is a program that gets my scheduling done. The code that follows runs on the mainframe under CMS and TSO, and on 'workframes' under OS/2. I have been toying with the issue of cross-system operability for many years. CLKQUEUE is one of my test beds for porting REXX code from one platform to another. Currently, this same code is being upgraded to run under LINUX.

I have always argued that IBM could easily have engineered their OS/2, VM, and MVS versions of REXX to be 99 percent the same. A few functions such as I/O and system routines could be in C, and nobody would be the wiser. But it's not to be. In fact the bare bones REXX won't even port well if the code has any meat in it. For instance, moving from VM to MVS the EXECIO commands don't support the same I/O capabilities. My feeling is if I can kludge it, and I'm a dummy, they should be able to build it. It's the little things that hurt most. For instance, under OS/2 the VM and MVS 'not' characters don't work, you have to resort to backslashes. The '@', '$', and '#' are not valid in tag names; host commands won't run unless prefaced by a 'CALL' command; and forget about EXECIO. Fortunately for me, I have licences for Personal REXX on DOS, Windows, and OS/2. Their REXX, at least, tries to live up to the mainframe standards REXX guys like myself are accustomed to.

Here's a piece of advice. Both MVS and VM REXX pass commands to the host system once it finds that it is not a REXX command. Under OS/2 you will get a chaining error if you try to do the same thing. I suggest, therefore, that developers get into the habit of coding host or operating system directed commands prefixed by the 'ALL' command. If the code is worth coding, there is a good chance that it will outlive MVS and VM – and then there will be one less activity at conversion time. There is one more thing you need to know about using the 'CALL' – RC won't work. Fortunately, 'RESULT' can be used instead. Check out the code. The 'CALL' is not elegant, but it's the only way to get portability.

Wouldn't it be nice if there was one, just one, language that could live up to what people really expect from an SAA language?

CODE NOTES

The three main components of CLKQUEUE follow. There are a number of sub-programs that are referenced. Most of them are cosmetic, and can be removed or replaced with simple home grown REXX programs. They cannot be included here because most of them have already been published in earlier issues of *VM Update*. For instance, CMSQ was published in issue 102 of *VM Update*, while REXXRDR and REXSAYIT were published in *VM Update* issues 93 and 94.

CLKQUEUE DOC

```
clkqueue ?

REXXNAME: CLKQUEUE

FUNCTION: TO QUEUE CMS/TSO/MVS/OS2/DOS COMMANDS AT THEIR APPOINTED
TIMES.

          1) RUNS CMS/TSO/MVS/OS2/DOS COMMANDS BASED ON "DATE" AND
             "TIME".
          2) RUNS CAN BE ONCE OR REQUEUED EVERY N NUMBER OF DAYS.
          3) CAN RERUN COMMANDS EVERY N HOURS, MINUTES, OR SECONDS.
          4) RERUNS CAN BE MADE TO STOP AT A CERTAIN TIME OR DAY.
          5) REPORTS ACTUAL RUN DATE, TIME, AND RET-CODE AFTER EACH RUN.
          6) RUNS CAN BE BASED ON PRIOR RETURN CODE SETTINGS USING IFS.
          7) UNLIKE SMART & PROFS ANY CMS/TSO OR EXEC/CLIST COMMAND CAN
             BE RUN.
          8) HAS SIMPLE ON-LINE DOCUMENTATION, AND FULL SET OF ERROR
             MESSAGES.
          9) RUNS SPECIAL TIME QUEUE SEQUENCES BY CALLING ITSELF.
         1Ø) CODE TIME BASED "IF" STATEMENTS TO HANDLE SPECIAL LOGICAL
             NEEDS.

HOWTORUN: ENTER COMMAND AS SHOWN BELOW.

          CLKQUEUE < &CTLQUE|* < &SLPMINS < &SLPCYCLS > > <*QUIET> >

          <      >  -MEANS THAT FIELDS WITHIN ARE OPTIONAL.
              |     -MEANS SELECT ONE OF TWO OR MORE OPTIONS.
           &CTLQUE  -ENTER THE NAME OF THE CLKQUEUE FILE.
                     THE FILETYPE MUST BE "CLKQUEUE".  IF "*" OR NO
                     PARAMETERS ARE ENTERED THEN THE DEFAULT FILENAM
                     OF "CLKQUEUE" IS USED.  IN "MVS" THIS CORRESPONDS
                     TO THE MEMBER NAME IN A DATASET NAMED...
          &SLPMINS  -ENTER THE NUMBER OF MINUTES CLKQUEUE SHOULD
```

```
                SLEEP AFTER EACH PASS THROUGH A QUEUE CLKQUEUE
                FILE.  IF "SLPMINS" IS ZERO OR NOT CODED THEN
                CLKQUEUE WILL PROCESS THE QUEUE CLKQUEUE FILE
              ONLY ONCE. THE MAXIMUM NUMBER OF SLEEP MINUTES IS 99.
      &SLPCYCLS -ENTER THE NUMBER OF TIMES CLKQUEUE SHOULD
                SLEEP THEN REPROCESS THE QUEUE CLKQUEUE FILE
                BEFORE IT STOPS RUNNING. THE DEFAULT IS ZERO.
        *QUIET   OPTIONAL KEYWORD THAT TELLS CLKQUEUE TO
                SUSPEND NORMAL MESSAGE PRINTING.  ERROR
                MESSAGES ARE STILL PRINTED.
      *ONEPASS   OPTIONAL KEYWORD.  IF CLKQUEUE QUEUES ONE
                OR MORE REQUESTS IT WILL REPROCESS THE INPUT
                CLOCK REQUESTS UNTIL IT FINDS NOTHING TO
                QUEUE.  THIS OPTION TURNS THE INPUT FILE
                REPROCESSING LOGIC OFF.
      *CLKRULES  OPTIONAL KEYWORD.  MEANS THAT INSTEAD OF
                EXECUTING QUEUED COMMANDS THEY ARE WRITTEN
                TO A FILE OF THE SAME FILENAME, BUT WITH THE
                FILETYPE OF "QUE".  THE FORMAT OF EACH OUTPUT
                RECORD IS:  QUETAG QUEDATE QUETIME QUEFACT
      *IFT(FT)   OPTIONAL KEYWORD.  USE THIS TO TELL CLKQUEUE
                TO USE AN INPUT FILE FILETYPE OF YOUR CHOICE,
                INSTEAD OF THE DEFAULT FILETYPE.

DESCRIBE: THIS EXEC WILL READ A FILE WITH THE FILETYPE OF CLKQUEUE
          SEARCHING FOR EXPIRED DATE AND TIMES.   THE FILENAME IS
          THE  USER'S  CHOICE,   BUT  IF  NO  FILENAME  IS  ENTERED
          THE DEFAULT IS CLKQUEUE. ONCE QUEUE DATE AND TIMES HAVE
          QUEUED NEW DATE AND TIMES ARE CREATED, THE ASSOCIATED
          COMMANDS ARE EXECUTED, AND INFO AND RETCODES SET BY THE
          COMMANDS ARE WRITTEN TO THE CLKQUEUE CONTROL FILE.  A FIELD
          CALLED THE QUEUE FACTOR IS USED TO CALCULATE WHEN TO
          SCHEDULE THE NEXT RUN, UNLESS IT'S ZERO, IN WHICH CASE IT
          IS RUN ONCE AND DROPPED.

QUEUEFMT: THIS PROGRAM USES A SCHEDULING CLKQUEUE FILE NAMED
          &QUECTL "CLKQUEUE" AS INPUT.  NOTE THAT &QUECTL IS A VARIABLE
          FOR WHICH "CLKQUEUE" IS DEFAULT.  IT SHOULD BE ON YOUR A DISK.
          IT CONTAINS THE RECORDS DEFINING WHAT TO SCHEDULE AND WHEN.
          BELOW IS A DEFINITION OF THE FIELDS IN EACH SCHEDULING
          RECORD.  RECORDS STARTING WITH AN ASTERISK(*) ARE TREATED
          AS COMMENTS, AND A RECORD STARTING WITH EOF SIGNALS THE
          END OF THE FILE.

  QUETAG QUEDATE QUETIME QUEFACT RUNDATE RUNTIME RUNCODE QUETEXT

          QUETAG  - TAG CAN UNIQUELY IDENTIFY THE COMMAND BEING RUN.
                    THE QUETAG IS ALSO USED AS A REXX VARIABLE INTO
                    WHICH THE QUETEXT RETURN CODE IS PLACED. TO SEE
```

```
                THE VALUES ASSOCIATED WITH ANY OF THE PREVIOUSLY
                RUN QUEUE ENTRIES PLEASE NOTE THAT THE ENTIRE
                RUN TIME QUEUE RECORD IS PUT INTO THE ZERO(Ø)
                INDEX VARIABLE OF QUETAG (IE QUETAG.Ø).
                FOR EXAMPLE TO CHECK THE DATE THAT CMDX WAS
                EXECUTED ENTER:
                    IF WORD(CMDX.Ø,5) = "9Ø/12/25" THEN "XMASLGC"
      QUEDATE - DATE OF NEXT RUN.  FORMAT = YY/MM/DD.
      QUETIME - TIME OF NEXT RUN.  FORMAT = HH/MM/SS.
      QUEFACT - INCREMENT FACTOR FOR CALCULATING NEXT RUN DATE/TIME.
                WHEN QUEFACT IS Ø COMMAND IS RUN ONLY ONCE.
         FMT = &DD|MONTH|YEAR<.H|M|S&NNN<*<XXX|HH:MM><.YY/MM/DD>>>
         &DD REPRESENTS THE NUMBER OF DAYS BEFORE COMMAND GETS
         REQUEUED.  FOR INSTANCE, SEVEN(7) MEANS RERUN COMMAND
         EVERY 7 DAYS. USE A PERIOD(.) TO INCLUDE A SECOND FIELD
         THAT SETS A CLKQUEUE TIME FACTOR.  ENTER 'MONTH' OR 'YEAR'
         TO REQUEUE MONTHLY AND YEARLY.   TO QUEUE A COMMAND USING
         A TIME FACTOR ENTER H FOR HOURS, M FOR MINUTES, OR S FOR
         SECONDS FOLLOWED BY THE TIME AMOUNT "NNN".  FOR A THREE
         MINUTE DELAY BETWEEN EXECUTIONS TO RUN EVERY DAY ENTER:
                    Ø1.MØØ3
                "XXX|HH:MM" REPRESENTS THE NUMBER OF TIMES THE TIME
                FACTOR SHOULD BE INCREMENTED OR THE HH:MM TIME
                THAT THE COMMAND SHOULD STOP RUNNING.  BOTH VALUES
                TOGETHER (NS&XS) CAN'T GO BEYOND MIDNIGHT.  TO RUN
                A COMMAND BETWEEN 8AM AND 4PM EVERY HOUR, ONCE
                PER WEEK CODE:
                    Ø8:ØØ:ØØ Ø7.H1*9  OR  Ø8:ØØ:ØØ Ø7.H1*16:ØØ
                ".YY/MM/DD" REPRESENTS THE STOP OR END DATE FOR
                THE QUEUE LOGIC.  TO MAKE THE ABOVE TEST CASE
                END ON APRIL 9TH 199Ø FOLLOW THE EXAMPLE SHOWN
                BELOW:
                    Ø8:ØØ:ØØ Ø7.H1*16:ØØ.9Ø/Ø4/Ø9
                NOTE.  CALCULATED STOP TIMES ARE RESET AFTER
                EACH EXECUTION OF THE LOOP OPTION (IE *XXX), UNLIKE
                EXPLICIT STOP TIMES (IE HH:MM).
    RUNDATE - DATE COMMAND WAS LAST RUN.  A Ø PLACE HOLDER IS REQUIRED.
    RUNTIME - TIME COMMAND WAS LAST RUN.  A Ø PLACE HOLDER IS REQUIRED.
    RUNCODE - RETCODE OF COMMAND LAST RUN.  A Ø PLACE HOLDER IS NEEDED.
              THIS VALUE CAN BE LOOKED AT WITH "IF" CODE BECAUSE
              AFTER EVERY RUN RETURN CODE IS PUT INTO ITS QUETAG VALUE.
    QUETEXT - TEXT OF COMMAND TO RUN.  ALL COMMANDS ARE ALLOWED.
             OPTIONALLY, "IF" STATEMENTS CAN EXECUTE
             QUEUED COMMANDS BY TESTING THE RETURN CODES OF PRIOR ONES.
             AN EXAMPLE IS SHOWN BELOW TO SEND MSG TO OPERATOR.
                    Ø IF CMD1 \= Ø THEN "CP MSG OP CMD1 AUTOLOG FAILED!"
                    IN CONDITIONALS USE "CLKQ" TO REUSE THIS LOGIC
                    WITH A UNIQUE OR SPECIALIZED QUEUE COMMAND SET.
                    IF THE WANTED COMMAND WILL NOT FIT ON THE QUEUE
                    RECORD, THE QUETEXT INFO MAY BE CODED ONTO THE
```

```
                    NEXT LINE BY PUTTING A COMMA AFTER THE RUNCODE.
                    NOTE, THE QUEUE PARAMETERS MUST ALL BE ON THE
                    SAME LINE, AND IF CONTINUATIONS ARE DONE NO
                    QUETEXT FIELDS CAN BE ON THE QUEUE INPUT LINE.
                    SUBSEQUENT CONTINUATIONS OF THE QUETEXT
                    FIELD MUST FOLLOW THE RULES OF REXX, AND ALL
                    CONTINUATION LINES, EXCEPT THE LAST, MUST END
                    WITH A COMMA(,) OR SEMI-COLON(;).
                    EX... IF CMDZ = 16 THEN "CLKQ EOYCYCL" ELSE NOP

CTLEXAMP: BELOW ARE SAMPLE CLKQUEUE RECORDS.

 CMD1 88/Ø1/22 14:3Ø:ØØ Ø1 ØØ/ØØ/ØØ ØØ:ØØ:ØØ Ø DIRLOG RSCS
 CMD2 88/Ø1/22 23:59:ØØ Ø1.M1Ø ØØ/ØØ/ØØ ØØ:ØØ:ØØ Ø CP QUERY RSCS
 88/Ø1/22 23:59:ØØ Ø1.M1Ø ØØ/ØØ/ØØ ØØ:ØØ:ØØ Ø IF CMD2=45 THEN MSG OP
RSCS DOWN!
 CMD4 88/Ø1/22 23:59:ØØ Ø1.M1Ø Ø Ø Ø IF CMD2 = 45 THEN DIRLOG RSCS
 CMDX: 88/Ø1/22 23:59:ØØ Ø1.M1Ø ØØ/ØØ/ØØ ØØ:ØØ:ØØ Ø,
       IF CMD1 \= Ø &,
          CMD2 \= Ø THEN DO;
           "MSG OP ************************";
           "MSG OP UNABLE TO RECOVER...";
           END
 EOF

CMDEXAMP: BELOW IS A REQUEST FOR CLKQUEUE TO KEEP TRYING TO QUEUE
          COMMANDS IN A FILE NAMED "TASKLIST CLKQUEUE" EVERY FIFTEEN
          MINUTES FOR 24 HOURS STRAIGHT.

          CLKQUEUE   TASKLIST   15   96

BATCHFMT: CLKQUEUE RUNS WELL WHEN SUBMITTED TO JES AS A BATCH JOB.  AN
          EXAMPLE OF THE JCL FOLLOWS, AND THE SYNTAX REMAINS THE SAME.
          REMEMBER, YOU MUST PUT YOUR CLKQUEUE CONTROLS INTO A DATASET
          CALLED 'YOURID.CLKQUEUE(CLKQUEUE)', AND SUBMIT IT FROM YOUR
          MATCHING TSO ID, FOR THIS JCL TO WORK.

          //CLOKSTEP EXEC PGM=IKJEFTØ1,DYNAMNBR=3Ø,REGION=4Ø96K,
          // PARM=('%CLKQUEUE 3 1Ø')
          //+*********************************************************
          //+ FOR DOCUMENTATION ENTER '%CLKQUEUE ?' IN THE PARM FIELD.
          //+*********************************************************
          //SYSEXEC  DD   DSN=MIRVI.REXX,DISP=SHR
          //SYSTSPRT DD   SYSOUT=*
          //SYSTSIN  DD   DUMMY

Ready; T=Ø.Ø8/Ø.22 14:17:Ø5
```

```
TEST RUN...
clkqueue clkqueue 1 3
IT WORKS.
CLKQUEUE - TEST: 96/12/23 14:32:00 1.M1 96/12/23 14:31:18 0 SAY 'IT
WORKS.'
CLKQUEUE - RAN 1 QUEUED COMMAND(S) ON 12/23/96 AT 14:31 FOR ID MIRVI.
CLKQUEUE - SLEEPING FOR 1 MINUTE(S) AT 14:31:19 ON 12/23/96.
IT WORKS.
CLKQUEUE - TEST: 96/12/23 14:33:00 1.M1 96/12/23 14:32:19 0 SAY 'IT
WORKS.'
CLKQUEUE - RAN 1 QUEUED COMMAND(S) ON 12/23/96 AT 14:32 FOR ID MIRVI.
CLKQUEUE - SLEEPING FOR 1 MINUTE(S) AT 14:32:19 ON 12/23/96.
IT WORKS.
CLKQUEUE - TEST: 96/12/23 14:34:00 1.M1 96/12/23 14:33:20 0 SAY 'IT
WORKS.'
CLKQUEUE - RAN 1 QUEUED COMMAND(S) ON 12/23/96 AT 14:33 FOR ID MIRVI.
CLKQUEUE - RAN A TOTAL OF 3 QUEUED COMMAND(S).
CLKQUEUE - 7 PASSES OF "CLKQUEUE CLKQUEUE" FILE MADE IN 3 QUEUEING
CYCLES.
Ready; T=9.41/9.80 14:33:20
spool cons close

CLKQUEUE CLKQUEUE

*
TEST: 96/12/23 14:34:00 1.M1 96/12/23 14:33:20 0 SAY 'IT WORKS.'
```

Editor's note: this article will be continued next month.

Marc Vincent Irvin
Move Immediate Software (USA)

Electronic bulletin board – part 4

This month we conclude the code for a bulletin board system.

EBBNEWS EXEC

```
/*  System     : EBBNEWS                                                 */
/*  EXEC name  : EBBNEWS                                                 */
/*  Invoked by : PROFS terminal user                                     */
/*  Function   : This EXEC displays the PROFS Bulletin Board menu.       */
  parse upper arg new
  'execio 1 cp (LIFO STRING Q SET'
  pull line
  wordnum = FIND(line,'EMSG')
  wordnum = wordnum + 1
  setemsg = WORD(line,wordnum)
  setemsg = STRIP(LEFT(setemsg,4))
  'cp set emsg off'
  'SET CMSTYPE HT'
  'MAKEBUF'
  'GETFMADR'
  pull . mode .
  'DROPBUF'
  'ACC 111' mode
  if rc <> Ø
     then do
          'CP LINK EBBNEWS 191 111 RR RNEWS'
          if rc <> Ø
             then do
                  'SET CMSTYPE RT'
                  'CP SET EMSG' setemsg
             say 'Error linking to EBBNEWS 191 (Bulletin Board disk).'
                  exit
                  end
          'VMFCLEAR'
          'ACC 111' mode
          if rc <> Ø
             then do
                  'SET CMSTYPE RT'
                  'CP SET EMSG' setemsg
              say 'Error accessing EBBNEWS 191 (Bulletin Board disk).'
                  exit
                  end
          end
  'SET CMSTYPE RT'
  naddr = '111'; nmode = mode
  'globalv select EBBNEWS put naddr nmode'
  if rc <> Ø
```

```
      then do
           'cp set emsg' setemsg
           exit rc
           end
   'set cmstype ht'
   'state $EBBNEWS $CONTROL A'
   saverc = rc
   'set cmstype rt'
   if saverc ¬= 0
      then do
           record = 'PRES     00000000 00:00'
           'execio 1 diskw $EBBNEWS $CONTROL A 0 F 80 (STRING' record
           record = 'SAFETY   00000000 00:00'
           'execio 1 diskw $EBBNEWS $CONTROL A (STRING' record
           record = 'VISITOR  00000000 00:00'
           'execio 1 diskw $EBBNEWS $CONTROL A (STRING' record
           record = 'VACATION 00000000 00:00'
           'execio 1 diskw $EBBNEWS $CONTROL A (STRING' record
           record = 'FYI      00000000 00:00'
           'execio 1 diskw $EBBNEWS $CONTROL A (STRING' record
           record = 'IDEAS    00000000 00:00'
           'execio 1 diskw $EBBNEWS $CONTROL A (STRING' record
           'finis $EBBNEWS $CONTROL A'
           end
   if new = 'N' | new ='NEW' then signal NEW_NEWS
   'exec EBBCHECK $OKAY$'
   'XEDIT $NEWS$ $MENU$ A (PROFILE EBBMENU'
   'REL 111'
   'cp set emsg' setemsg
   'vmfclear'
   exit
NEW_NEWS:
   'execio * diskr $EBBNEWS $CONTROL A (STEM' ctlrec.
   'finis $EBBNEWS $CONTROL A'
   if ctlrec.0 = 0 then signal DONE
   count = 0
   i = 0
   do ctlrec.0
      i = i + 1
      board = SUBSTR(ctlrec.i,1,8)
      bdate = SUBSTR(ctlrec.i,10,8)
      btime = SUBSTR(ctlrec.i,19,5)
      brest = SUBSTR(ctlrec.i,25,55)         /* ' ' or 'Screened'      */
      parse upper var brest brest
      if WORD(brest,1) = 'SCREENED'
         then iterate
      bcomp = bdate || btime
      'execio * diskr' board 'EBBNEWS' nmode '(STEM' item.
      j = 1
      do until j > item.0
         if j > 2
```

```
           then do
                idate = SUBSTR(item.j,101,8)
                itime = SUBSTR(item.j,75,5)
                icomp = idate || itime
                if icomp > bcomp
                   then do
                        count = count + 1
                        end
                end
        j = j + 1
     end
  end
  if count = 0
     then signal NO_NEWS
  if count = 1
     then do
          queue 'MSG There is 1 new bulletin item.  Press PF8 to view.'
          signal YES_NEWS
          end
  if count > 1
     then do
  queue 'MSG There are' count 'new bulletin items.  Press PF8 to view.'
          signal YES_NEWS
          end
YES_NEWS:
  'XEDIT $NEWS$ $MENU$ A (PROFILE EBBMENU'
  'set cmstype ht'
  'REL 111'
  'cp set emsg' setemsg
  'set cmstype rt'
  exit
NO_NEWS:
  'set cmstype ht'
  'REL 111'
  'cp set emsg' setemsg
  'set cmstype rt'
  'SUBCOM XEDIT'
  if rc = 0 & not cmsflag('SUBSET')
     then address 'XEDIT',
        'MSG There are no new items on any "unscreened" bulletin boards'
     else
        say 'There are no new items on any "unscreened" bulletin boards'
  exit
```

EBBNEWS XEDIT

```
/*  System     : EBBNEWS                                                */
/*  Macro name : EBBNEWS                                                */
/*  Invoked by : EBBNEWS EXEC                                           */
/*  Function   : This macro formats the PROFS Bulletin Board Menu.      */
```

```
/*                                                                    */
address CMS 'GLOBALV SELECT EBBNEWS GET naddr nmode'
'COMMAND SET AUTOSAVE OFF'
'COMMAND SET MSGMODE OFF'
'COMMAND SET SCOPE ALL'
'COMMAND SET CASE M I'
'COMMAND SET CMDLINE OFF'
'COMMAND SET CURLINE ON 17'
'COMMAND SET MSGLINE ON 12'
'COMMAND SET PREFIX OFF'
'COMMAND SET SCALE OFF'
'COMMAND SET TOFEOF OFF'
'COMMAND SET WRAP ON'
'COMMAND SET STAY ON'
'COMMAND SET SHADOW OFF'
'COMMAND SET COLOR  *        BLUE   NONE    HIGH'
'COMMAND SET COLOR  CURLINE  YELLOW NONE  NOHIGH'
'COMMAND SET COLOR  FILEAREA YELLOW NONE  NOHIGH'
'COMMAND SET COLOR  MSGLINE  RED    NONE    HIGH'
'COMMAND SET LINEND OFF'
'COMMAND SET ENTER EXEC EBBCHECK $OKAY$'
'COMMAND SET PFØ1 EXEC EBBBULL $OKAY$ PRES'
'COMMAND SET PFØ2 EXEC EBBBULL $OKAY$ SAFETY'
'COMMAND SET PFØ3 EXEC EBBBULL $OKAY$ VISITOR'
'COMMAND SET PFØ4 EXEC EBBBULL $OKAY$ VACATION'
'COMMAND SET PFØ5 EXEC EBBBULL $OKAY$ FYI'
'COMMAND SET PFØ6 EXEC EBBBULL $OKAY$ IDEAS'
'COMMAND SET PFØ7 EXEC EBBCURR $OKAY$ ALL'
'COMMAND SET PFØ8 EXEC EBBCURR $OKAY$ NEW'
'COMMAND SET PFØ9 EXEC EBBHELP EBBMENU'
'COMMAND SET PF1Ø EXEC EBBHELP EBBMENU'
'COMMAND SET PF11 EBBCREEN'
'COMMAND SET PF12 QQUIT'
'COMMAND SET LINEND ON #'
'COMMAND SET CTLCHAR ! ESCAPE'
'COMMAND SET CTLCHAR % PROTECT HIGH'
'COMMAND SET CTLCHAR @ PROTECT NOHIGH'
xxx   = DATE(W) '     '
yyy   = 'Electronic News - PROFS Bulletin Board'
zzz   = '    ' DATE(USA)
'COMMAND SET RESERVED  1 N  !@' xxx ' !%' yyy ' !@' zzz
'COMMAND SET RESERVED  2 BLUE NONE N  '
xxx   = '  Press one of the following PF keys.'
'COMMAND SET RESERVED  3 BLUE NONE N' xxx
'COMMAND SET RESERVED  4 BLUE NONE N  '
address CMS 'EXECIO * DISKR $EBBNEWS $CONTROL A (STEM' ctlrec.
address CMS 'FINIS $EBBNEWS $CONTROL A'
xxx = '  PF1'
yyy = 'From The Desk of... Steve Bruntlett'
zzz  = WORD(ctlrec.1,4)
if zzz <> '' then zzz = '.....................'zzz
```

```
'COMMAND SET RESERVED  5 WHITE NONE HIGH' xxx '!@'yyy zzz
xxx = '  PF2'
yyy = 'REPORTS and Safety News'
zzz  = WORD(ctlrec.2,4)
if zzz <> '' then zzz = '.........................'zzz
'COMMAND SET RESERVED  6 WHITE NONE HIGH' xxx '!@'yyy zzz
xxx = '  PF3'
yyy = 'Vistors and Plant Tours'
zzz  = WORD(ctlrec.3,4)
if zzz <> '' then zzz = '.........................'zzz
'COMMAND SET RESERVED  7 WHITE NONE HIGH' xxx '!@'yyy zzz
xxx = '  PF4'
yyy = 'Vacations and OUT-OF-TOWN Notices'
zzz  = WORD(ctlrec.4,4)
if zzz <> '' then zzz = '................'zzz
'COMMAND SET RESERVED  8 WHITE NONE HIGH' xxx '!@'yyy zzz
xxx = '  PF5'
yyy = 'FYI - General Information'
zzz  = WORD(ctlrec.5,4)
if zzz <> '' then zzz = '........................'zzz
'COMMAND SET RESERVED  9 WHITE NONE HIGH' xxx '!@'yyy zzz
xxx = '  PF6'
yyy = 'Task Force Groups'
zzz  = WORD(ctlrec.6,4)
if zzz <> '' then zzz = '.............................'zzz
'COMMAND SET RESERVED 1Ø WHITE NONE HIGH' xxx '!@'yyy zzz
'COMMAND SET RESERVED 11 BLUE NONE N   '
'COMMAND SET RESERVED 12 BLUE NONE N   '
'COMMAND SET RESERVED 13 BLUE NONE N   '
yyy = '  PF7 All!@Bulletins  !%PF8 New!@Bulletins  !%PF9!@Help '
zzz = '!%PF11!@"Screen"  !%PF12!@End'
'COMMAND SET RESERVED 14 WHITE NONE HIGH' yyy zzz
xxx = COPIES('-',79)
'COMMAND SET RESERVED 15 BLUE NONE HIGH' xxx
'COMMAND SET RESERVED 16 BLUE NONE HIGH  '
'COMMAND SET RESERVED 17 BLUE NONE HIGH  '
'COMMAND SET RESERVED 18 BLUE NONE HIGH  '
'COMMAND SET RESERVED 19 BLUE NONE HIGH  '
'COMMAND SET RESERVED 2Ø BLUE NONE HIGH  '
'COMMAND SET RESERVED 21 BLUE NONE HIGH  '
'COMMAND SET RESERVED 22 BLUE NONE HIGH  '
'COMMAND SET RESERVED 23 BLUE NONE HIGH  '
'COMMAND SET RESERVED 24 BLUE NONE HIGH  '
'COMMAND EXTRACT /LSCREEN'
if LSCREEN.5 = 27
   then do
        'COMMAND SET RESERVED 25 BLUE NONE HIGH  '
        'COMMAND SET RESERVED 26 BLUE NONE HIGH  '
        'COMMAND SET RESERVED 27 BLUE NONE HIGH  '
        end
'COMMAND SET MSGMODE ON'
```

```
'COMMAND CURSOR SCREEN 1 1'
exit
```

EBBPRINT XEDIT

```
/*  System     : EBBNEWS                                                 */
/*  Macro name : EBBPRINT                                                */
/*  Invoked by : EBBBULL XEDIT macro                                     */
/*  Function   : This macro prints detail information on an item         */
/*               listed in the PROFS Bulletin Board.                     */
  'GLOBALV SELECT EBBNEWS GET naddr nmode'
EXTRACT:
  'COMMAND EXTRACT /CURSOR/LSCREEN/SIZE/LINE/FNAME'
  cursscrn = CURSOR.1
  cursfile = CURSOR.3
  screenl  = LSCREEN.1
  filel    = SIZE.1
  saveline = LINE.1
  if cursscrn = 1
     then signal PRINT_BOARD
CHECK:
  if cursscrn > screenl  then signal MSG
  if cursscrn < 2        then signal MSG
  if cursfile > filel    then signal MSG
  if cursfile < 1        then signal MSG
GETFN:
  'COMMAND :' cursfile
  'COMMAND EXTRACT /CURLINE'
  'COMMAND :' saveline
  strg = CURLINE.3
  if strg = ' ' then exit
  more = SUBSTR(strg,52,8)
  if more = '        '
     then do
          'COMMAND EMSG There is no more information on this item.'
          exit
          end
  fn   = SUBSTR(strg,81,8)
  ft   = SUBSTR(strg,9Ø,8)
  type = SUBSTR(strg,99,1)
PRINT_ITEM:
  'SET CMSTYPE HT'
  'STATE' fn ft nmode
  saverc = rc
  'SET CMSTYPE RT'
  if saverc <> Ø
     then do
          'COMMAND EMSG No reference document found on this item.'
          exit
          end
```

```
  'COPYFILE' fn ft nmode '$EBBNEWS SCRIPT A (REPLACE'
  if type = 'N' then 'PROFS HARDCOPY $EBBNEWS SCRIPT A'
  if type = 'P' then 'PROFS MEMO $EBBNEWS SCRIPT A'
  exit
MSG:
  'COMMAND EMSG Place the cursor next to the item to view',
     'and press ENTER or PF4...'
  exit
PRINT_BOARD:
  board = FNAME.1
  xxx   = ' 'DATE(W) '            '
  yyy   = 'Electronic News - PROFS Bulletin Board'
  zzz   = '        ' DATE(USA)
  rec   = xxx yyy zzz
  'ERASE $EBBNEWS SCRIPT A'
  'EXECIO 1 DISKW $EBBNEWS SCRIPT A 1 F 8Ø (STRING' rec
  'EXECIO 2 DISKW $EBBNEWS SCRIPT A 2 (STRING     '
  'FINIS $EBBNEWS SCRIPT A'
  if board = '$EBBNEWS'
     then 'COPYFILE $EBBNEWS $CURRENT A $EBBNEWS SCRIPT A (APPEND'
     else 'COPYFILE' board 'EBBNEWS' nmode '$EBBNEWS SCRIPT A (APPEND'
  'XEDIT $EBBNEWS SCRIPT A (PROFILE EBBPRTX'
  'PROFS HARDCOPY $EBBNEWS SCRIPT A'
  'ERASE $EBBNEWS SCRIPT A'
  exit
```

EBBPINTX XEDIT

```
/*  System     : EBBNEWS                                              */
/*  Macro name : EBBPRTX                                              */
/*  Invoked by : EBBPRINT xedit macro                                 */
/*  Function   : This XEDIT macro edits the $EBBNEWS SCRIPT file that */
/*               is built for printing the bulletin boards.           */
  'COMMAND DOWN 2'
  'COMMAND SHIFT RIGHT 1 *'
  'COMMAND TOP'
  'COMMAND INPUT .fo off'
  'COMMAND FILE'
  exit
```

EBBRECV EXEC

```
/*  System     : EBBNEWS                                              */
/*  EXEC name  : EBBRECV                                              */
/*  Invoked by : EBBSTART EXEC                                        */
/*  Function   : This EXEC receives a message from a PROFS user's     */
/*               virtual machine to either add, chg, or del items     */
/*               from a bulletin and EBBNEWS' A disk.                 */
/*  read command as received by the secretary                         */
```

```
  parse upper arg message
  parse var message usr cmd board a1 a2 a3 a4 .
  if cmd = ':::' then signal EXIT
  if cmd = 'CP' then signal CP_COMMAND
  usr = SUBSTR(usr,1,8)
/*  check to make sure the user is authorized to EBBNEWS              */
  'DESBUF'
  'EXECIO * DISKR EBBUSER AUTH A0 (FINIS ZONE 1 8 LOCATE /'usr'/'
  if rc > 0
     then do
          text = 'NOT AUTHORIZED TO USE ANY EBBNEWS COMMANDS'
          signal SEND_REPLY
          end
  if QUEUED() <> 2
     then do
          text = 'ERROR LOCATING USER : ' usr
          signal SEND_REPLY
          end
  parse pull recnum; parse pull record
  parse var record . uinit upswd .
/*  branch to the appropriate command processing                      */
BRANCH:
  signal VALUE cmd
  text = 'ERROR IN EBBRECV EXEC... COMMAND VALUE ERROR'
  signal SEND_REPLY
ADD:
  'GLOBALV SELECT EBBNEWS GET counter'
  counter = counter + 1
  'GLOBALV SELECT EBBNEWS PUTP counter'
  'DESBUF'
  'GETFMADR 200'
  parse pull . user_mode user_vaddr .
  'CP LINK' usr '191' user_vaddr 'RR' upswd
  'ACC' user_vaddr user_mode
  'COPYFILE $EBBNEWS $SUBJ$' user_mode 'EBBNEWS SUBJECT A (REPLACE'
  'EXECIO 1 DISKR EBBNEWS SUBJECT A (FINIS STEM' rec.
  readrc = rc
  if readrc <> 0
     then do
          'REL' user_vaddr '(DET'
          text = 'EBBNEWS cannot read the subject from your A disk.',
                 'ERROR=' readrc
          signal SEND_REPLY
          end
  subj = SUBSTR(rec.1,1,50)
  subj = STRIP(subj,,'_')
  subj = SUBSTR(subj,1,50)
  'STATE $EBBNEWS SCRIPT' user_mode
  if rc <> 0
     then do
          more = '        '
```

```
          fn = RIGHT(counter,8,'Ø')
          ft = '........'
          signal ADD2
          end
  more = 'more... '
  fn   = RIGHT(counter,8,'Ø')
  ft   = 'SCRIPT  '
  'COPYFILE $EBBNEWS SCRIPT' user_mode fn 'SCRIPT A'
  copyrc = rc
  if copyrc <> Ø
     then do
          'REL' user_vaddr '(DET'
     text = 'ERROR' copyrc 'ADDING YOUR ITEM TO THE EBBNEWS A DISK.',
            'CALL SYSTEMS.'
          signal SEND_REPLY
          end
ADD2:
  'REL' user_vaddr '(DET'
  init = SUBSTR(uinit,1,4)
  dte  = DATE(USA)
  dte2 = DATE(S)
  tme  = TIME()
  tme  = SUBSTR(tme,1,5)
  flag = 'N'
  if a1 = 'PROFS'
     then flag = 'P'
  record = subj more init dte tme fn ft flag dte2
  'EXECIO 1 DISKW' board 'EBBNEWS A Ø F 11Ø (FINIS STRING' record
  if rc > Ø
     then do
     text = 'ERROR WRITING TO THE BULLETIN BOARD... CONTACT SYSTEMS'
          signal SEND_REPLY
          end
  if a4 = 'Y' then call FLASH
  text = 'OKAY'
  signal SEND_REPLY
DELETE:
  queue 'EBBDELX' uinit a2 a3
  'XEDIT' board 'EBBNEWS A'
  saverc = rc
  if saverc <> Ø
     then do
          text = 'ERROR DELETING BULLETIN ITEM FOR' uinit a2 a3
          signal SEND_REPLY
          end
  text = 'OKAY'
  signal SEND_REPLY
REPLACE:
  'GLOBALv select EBBNEWS get counter'
  counter = counter + 1
  'GLOBALv select EBBNEWS putp counter'
```

```
  'DESBUF'
  'GETFMADR 200'
  parse pull . user_mode user_vaddr .
  'CP LINK' usr '191' user_vaddr 'RR' upswd
  'ACC' user_vaddr user_mode
  'COPYFILE $EBBNEWS $SUBJ$' user_mode 'EBBNEWS SUBJECT A (REPLACE'
  'EXECIO 1 DISKR EBBNEWS SUBJECT A (FINIS STEM' rec.
  readrc = rc
  if readrc <> 0
     then do
          'REL' user_vaddr '(DET'
     text = 'EBBNEWS cannot read the subject from your A disk.',
            'ERROR=' readrc
          signal SEND_REPLY
          end
  subj = SUBSTR(rec.1,1,50)
  subj = STRIP(subj,,'_')
  subj = SUBSTR(subj,1,50)
  'STATE $EBBNEWS SCRIPT' user_mode
  if rc <> 0
     then do
          more = '        '
          fn = RIGHT(counter,8,'0')
          ft = '........'
          signal REPL2
          end
  more = 'more... '
  fn   = RIGHT(counter,8,'0')
  ft   = 'SCRIPT  '
  'COPYFILE $EBBNEWS SCRIPT' user_mode fn 'SCRIPT A'
  copyrc = rc
  if copyrc <> 0
     then do
          'REL' user_vaddr '(DET'
     text = 'ERROR' copyrc 'ADDING YOUR ITEM TO THE EBBNEWS A DISK.',
            'CALL SYSTEMS.'
          signal SEND_REPLY
          end
REPL2:
  'REL' user_vaddr '(DET'
  if a2 <> '........'
     then do
          queue 'EBBDELX' uinit a2 a3
          'XEDIT' board 'EBBNEWS A'
          end
  init = SUBSTR(uinit,1,4)
  dte  = DATE(USA)
  dte2 = DATE(S)
  tme  = TIME()
  tme  = SUBSTR(tme,1,5)
  flag = 'N'
```

```
  if a1 = 'PROFS'
     then flag = 'P'
  record = subj more init dte tme fn ft flag dte2
  'EXECIO 1 DISKW' board 'EBBNEWS A Ø F 11Ø (FINIS STRING' record
  if rc > Ø
     then do
     text = 'ERROR WRITING TO THE BULLETIN BOARD... CONTACT SYSTEMS'
          signal SEND_REPLY
          end
  if a4 = 'Y' then call FLASH
  text = 'OKAY'
  signal SEND_REPLY
/*  Send a news flash of the bulletin item if requested by author     */
FLASH:
  hi = '1DE8'X
  lo = '1D6Ø'X
  blnk = lo || ' '
  msg1 = hi || 'PROFS Bulletin Board News Flash...' || lo
  msg2 = lo || SUBSTR(record,1,7Ø) SUBSTR(record,75,5)
  'MAKEBUF'
  'QUERY LINKS 399 (FIFO'
  'SENTRIES'
  indx = rc
  i = Ø
  do indx
      i = i + 1
      parse upper pull usr.1 . . usr.2 . . usr.3 . . usr.4 . .
      do n = 1 to 4
         if usr.n || '' then call SENDIT
         end
  end
  'DESBUF'
  return
SENDIT:
  if usr.n = 'SYSADMIN' then return
  if usr.n = 'EBBNEWS'  then return
  if usr.n = 'EBBCAL'   then return
  if usr.n = 'EBBDBM'   then return
  if usr.n = 'EBBMAIL'  then return
  'EXECIO * CP (STRING MSGNOH' usr.n msg1
  'EXECIO * CP (STRING MSGNOH' usr.n msg2
  return
/*  intercept cp command requests and route output back to requestor  */
CP_COMMAND:
  wrd = FIND(message,'CP')
  loc = WORDINDEX(message,wrd)
  len = LENGTH(message)
  len = len - loc
  cpcmd = SUBSTR(message,loc,len)
  if cpcmd = 'Q TERM' | cpcmd = 'QUERY TERM' then signal EXIT
  'DESBUF'
```

```
  'EXECIO * CP (FIFO STRING' cpcmd
  do QUEUED()
     pull text
     'CP MSGNOH' usr text
  end
  'DESBUF'
  signal EXIT
/*  send a message back to the invoker                                 */
SEND_REPLY:
  'CP SMSG' usr text
EXIT:
  exit
```

EBBREPL EXEC

```
/*  System     : EBBNEWS                                               */
/*  EXEC name  : EBBREPL                                               */
/*  Invoked by : EBBREPLX XEDIT macro                                  */
/*  Function   : This EXEC replaces detail information on a Bulletin  */
/*               Board.  The user is prompted for either the cron      */
/*               number or a filename/filetype or data entered is      */
/*               saved as $TEMP$ $DATA$.  The REPLACE command is sent */
/*               to EBBNEWS for actual delete and re-add.              */
  parse arg okay board rest
  if okay <> '$OKAY$'
     then do
          say 'This EXEC can only be invoked through the EBBNEWS EXEC'
          exit 99
          end
  'GLOBALV SELECT EBBNEWS GET naddr nmode'
  'SET CMSTYPE HT'
  'ACC' naddr nmode
  saverc = rc
  'SET CMSTYPE RT'
  if saverc then exit rc
  'VMFCLEAR'
  'SET CMSTYPE HT'
  'STATE $EBBNEWS SCRIPT A'
  if rc = Ø then 'ERASE $EBBNEWS SCRIPT A'
  'STATE $EBBNEWS $SUBJ$ A'
  if rc = Ø then 'ERASE $EBBNEWS $SUBJ$ A'
  'STATE $EBBNEWS $JUNK$ A'
  if rc = Ø then 'ERASE $EBBNEWS $JUNK$ A'
  'SET CMSTYPE RT'
  queue 'REPLACE' board rest
  'XEDIT $EBBJUNK SCRIPT A (PROFILE EBBEDIT'
  'ERASE $EBBNEWS $SUBJ$ A'
  'ERASE $EBBNEWS $JUNK$ A'
  'VMFCLEAR'
  queue 'QQUIT'
  exit
```

EBBREPLX XEDIT

```
/*  System     : EBBNEWS                                                 */
/*  Macro name : EBBREPLX                                                */
/*  Invoked by : EBBBULL XEDIT macro                                     */
/*  Function   : This macro gets the item description and invokes the */
/*               EBBREPL EXEC.                                           */
  parse arg board desc
  'COMMAND EXTRACT /CURSOR/LSCREEN/SIZE/LINE'
  cursscrn = CURSOR.1
  cursfile = CURSOR.3
  screenl  = LSCREEN.1
  filel    = SIZE.1
  saveline = LINE.1
  if cursscrn > screenl  then signal MSG
  if cursscrn < 2        then signal MSG
  if cursfile > filel    then signal MSG
  if cursfile < 1        then signal MSG
  'CP SET SMSG ON'
  'SMSG EBBNEWS ::: '
  if rc = 45
     then do
  'EMSG EBBNEWS is not logged on... check with computer operations'
          'CP SET SMSG OFF'
          exit
          end
     else
     if rc = 57
        then do
  'EMSG EBBNEWS is not receiving... check with computer operations'
             'CP SET SMSG OFF'
             exit
             end
     else
     if rc > Ø
        then do
  'EMSG EBBNEWS is not available... NOTIFY OPERATIONS IMMEDIATELY'
             'CP SET SMSG OFF'
             exit
             end
  'CP SET SMSG OFF'
GETFN:
  'CMS GLOBALV SELECT EBBNEWS GET naddr nmode'
  'COMMAND :' cursfile
  'COMMAND EXTRACT /CURLINE'
  'COMMAND :' saveline
  strg = CURLINE.3
  if strg = ' ' then exit
  if SUBSTR(strg,1,5) = '-----' then exit
  init = SUBSTR(strg,61,3)
  'DESBUF'
```

```
  'EXECIO * DISKR EBBUSER LIST' nmode,
                               '(FINIS ZONE 1Ø 12 LOCATE /'init'/'
  if rc > Ø
     then do
          'EMSG You are not authorized to replace Bulletin items'
          exit
          end
  if QUEUED() <> 2
     then do
          'EMSG Error reading user list.  Call systems'
          exit
          end
  parse pull recnum; parse pull record
  parse var record ruser rinit rusr1 rusr2 rusr3 .
  usr = USERID()
  if usr = ruser | usr = rusr1 | usr = rusr2 | usr = rusr3
     then nop
     else do
  "EMSG You are not authorized to replace" ruser || "'s bulletins"
          exit
          end
  bull = SUBSTR(strg,1,5Ø)
  fn   = SUBSTR(strg,81,8)
  ft   = SUBSTR(strg,9Ø,8)
  type = SUBSTR(strg,99,1)
EXECUTE:
  'EXEC EBBREPL $OKAY$' board fn ft bull ':::' desc
  if rc = Ø
     then do
          'COMMAND :' cursfile
          'COMMAND CLOCATE :1'
          msg = LEFT('------ replaced ------',8Ø,' ')
          'COMMAND CREPLACE' msg
          'COMMAND :' saveline
          end
  'COMMAND CURSOR SCREEN' cursscrn '1'
  exit
MSG:
  'EMSG Place the cursor next to the item to replace and press PF2'
  exit
```

EBBSEND EXEC

```
/*  System     : EBBNEWS                                              */
/*  EXEC name  : EBBSEND                                              */
/*  Invoked by : EBBADD, EBBCHG and EBBDEL XEDIT macros               */
/*  Function   : This EXEC sends a request to the EBBNEWS machine    */
/*               for adding, deleting, or changing bulletin board     */
/*               items.                                               */
```

```
/*                                                                    */
  a1 = ''; a2 = ''; a2 = ''; a2 = ''
  parse arg okay cmd board a1 a2 a3 a4 .
  if okay <> '$OKAY$'
     then do
          say 'This EXEC can only be invoked through the EBBNEWS EXEC'
          exit 99
          end

  'CP SET SMSG ON'
  message = cmd board a1 a2 a3 a4 ':::'
  'SMSG EBBNEWS' message
  'WAKEUP (SMSG'
  smsg_rc = rc
  if smsg_rc = 6
     then do
          'VMFCLEAR'
          say 'If you interrupt EBBNEWS (by pressing a key) while it is'
          say 'running you will cause the' cmd 'in progress to abend!  '
          'WAKEUP (SMSG'
          smsg_rc = rc
          end
  'CP SET SMSG OFF'
  if smsg_rc <> 1 then exit 99
  parse pull reply
  parse var reply . . status .
  if status <> 'OKAY'
     then do
          'VMFCLEAR'
          reply = SUBSTR(reply,18,LENGTH(reply))
          say reply
          say ' '
          say 'Press Enter to continue...'
          say ' '
          parse pull dummy
          exit 99
          end
  exit
```

EBBSTART EXEC

```
/*  System     : EBBNEWS                                              */
/*  EXEC name  : EBBSTART                                             */
/*  Invoked by : EBBNEWS' PROFILE exec                                */
/*  Function   : This EXEC sets the EBBNEWS virtual machine up for    */
/*               receiving messages (commands) and passing the data   */
/*               to the EBBRECV EXEC.                                 */
'VMFCLEAR'
'SET CMSTYPE HT'
'EXECDROP EBBRECV EXEC'
```

```
'EXECLOAD EBBRECV EXEC'
'SET CMSTYPE RT'
say 'EBBNEWS NOW EXECUTING...'
say ''
say 'type #CP DISC to let EBBNEWS run or press ENTER twice to end'
'CP SET SMSG ON'
WAKEUP:
/*'WAKEUP (FILE (EBBFLASH TIMES A SMSG'*/
'WAKEUP (SMSG'
if rc = 1                    /* smsg */
   then do
/*      pull message                    needed for EBBflash if used */
        pull message
        'EBBRECV' SUBSTR(message,7,LENGTH(message))
        signal WAKEUP
        end
if rc = 3                    /* time */
   then do
        'EBBFLASH'
        signal WAKEUP
        end
'CP SET SMSG OFF'
exit
```

EBBUSER LIST

```
USERIDØ1 XXX  BACKUPØ1 BACKUPØ2 BACKUPØ3
USERIDØ2 XXX  BACKUPØ1 BACKUPØ2 BACKUPØ3
USERIDØ3 XXX  BACKUPØ1 BACKUPØ2 BACKUPØ3
USERIDØ4 XXX  BACKUPØ1 BACKUPØ2 BACKUPØ3
USERIDØ5 XXX  BACKUPØ1 BACKUPØ2 BACKUPØ3
```

EBBVIEW XEDIT

```
/*  System     : EBBNEWS                                              */
/*  Macro name : EBBVIEW                                              */
/*  Invoked by : EBBBULL XEDIT macro                                  */
/*  Function   : This macro displays detailed information on an item  */
/*               listed in the PROFS Bulletin Board.                  */
  'GLOBALV SELECT EBBNEWS GET naddr nmode'
EXTRACT:
  'COMMAND EXTRACT /CURSOR/LSCREEN/SIZE/LINE'
  cursscrn = CURSOR.1
  cursfile = CURSOR.3
  screenl  = LSCREEN.1
  filel    = SIZE.1
  saveline = LINE.1
CHECK:
  if cursscrn > screenl  then signal MSG
```

```
  if cursscrn < 2        then signal MSG
  if cursfile > filel    then signal MSG
  if cursfile < 1        then signal MSG
GETFN:
  'COMMAND :' cursfile
  'COMMAND EXTRACT /CURLINE'
  'COMMAND :' saveline
  strg = CURLINE.3
  if strg = ' ' then exit
  more = SUBSTR(strg,52,8)
  if more = '        '
     then do
          'COMMAND EMSG There is no more information on this item.'
          exit
          end
  fn   = SUBSTR(strg,81,8)
  ft   = SUBSTR(strg,90,8)
  type = SUBSTR(strg,99,1)
EXECUTE:
  'COMMAND SET CMSTYPE HT'
  'STATE' fn ft nmode
  saverc = rc
  'COMMAND SET CMSTYPE RT'
  if saverc <> 0
     then do
          'COMMAND EMSG No reference document found on this item.'
          exit
          end
  'COPYFILE' fn ft nmode '$EBBNEWS SCRIPT A (REPLACE'
  if type = 'N' then 'PROFS OFSMOSCR $EBBNEWS SCRIPT A'
  if type = 'P' then 'PROFS MEMO $EBBNEWS SCRIPT A'
  exit
MSG:
  'COMMAND EMSG Place the cursor next to the item to view',
              'and press ENTER or PF4'
  exit
```

INITGL EXEC

```
/* initgl EXEC to initialize the global counter*/
  counter = 00000000
  'globalv select EBBNEWS putp counter'
```

P C Shumway
Systems Analyst (USA)

SFS directory listing for all directories in a filepool

OVERVIEW

CMS offers the command DIRLIST to display all the directories beneath a root directory. Sometimes, however, it would be preferable to see all the directories of all the filepools within one display. The DLALL EXEC was created to meet this requirement.

SYNTAX

DLALL is called with the following parameters:

```
DLALL   <sort (Refresh>
```

'sort' defines the sort sequence of the display entries. MODE sorts by filemode (showing the currently accessed directories sorted by filemode before the non-accessed directories), and DIR (the default) sorts by directory name. This option is only effective in conjunction with the following refresh option.

If '(Refresh' or '(R' is specified, the directory structure is examined again; without this option the structure of the last DLALL call is displayed. A refresh is not necessary when you know that the directory structure has not changed. This is a performance issue. If there was no previous DLALL call (indicated by file 'userid DIRLALL A' not being present), the refresh option is assumed.

'?' gives a help panel for the function.

LIMITATIONS

At the moment the filepools VMSYS: and VMSYSU: are worked on. You can easily enhance the EXEC to cover other filepools.

INTERNALS AND PREREQUISITES

The calling user must have the authority to query all enrolled users of the filepools to find all root directories.

To show the results, profile DL XEDIT is used. It calls the original IBM profile PROFDLST XEDIT.

INSTALLATION-SPECIFIC CONFIGURATION

The temporary file that holds the results is on the user's A disk as file name 'userid DIRLALL' where 'userid' is the CMS user-id. This is a hardcoded value.

OUTPUT PRESENTED BY DLALL

The results of a 'DLALL (R' command are shown in Figure 1.

```
 MAINT    DIRLIST  A0  V 319  Trunc=319 Size=387 Line=1 Col=1 Alt=1
Cmd   Fm Directory Name
      -  VMSYS:MAINT.
      -  VMSYSU:BARR.
      -  VMSYSU:BARR.GGT
      -  VMSYSU:BARR.SAVE
      -  VMSYSU:DOS1.
      -  VMSYSU:DOS1.SAVE
      -  VMSYSU:DOSP.
      -  VMSYSU:DOSP.SAVE
      C  VMSYSU:DOST.
      -  VMSYSU:DOST.BSP
      -  VMSYSU:DOST.CCC
      M  VMSYSU:DOST.MAT
      -  VMSYSU:DOST.PTT
      -  VMSYSU:DOST.PVT
      -  VMSYSU:DOST.SAVE
      D  VMSYSU:DOST.TSTC1
      T  VMSYSU:DOST.TSTC2
1= Help       2= Refresh  3= Quit   4= Sort(fm)    5= Sort(dir)   6= Auth
7= Backward   8= Forward  9=       10=            11= Filelist   12= Cursor

====>
                                                          X E D I T  1 File
```

Figure 1: Example output

The results of a ‘DLALL MODE (R’ command are shown in Figure 2.

```
 MAINT    DIRLIST  A0  V 319  Trunc=319 Size=387 Line=1 Col=1 Alt=4
Cmd   Fm Directory Name
      C  VMSYSU:DOST.
      D  VMSYSU:DOST.TSTC1
      F  VMSYSU:KL.JCL
      H  VMSYSU:MAINT.
      I  VMSYSU:KL.
      J  VMSYSU:KL.DIS.TEST
      K  VMSYSU:KL.MAT.KK
      L  VMSYSU:KL.MAT
      M  VMSYSU:DOST.MAT
      N  VMSYSU:KL.SYS
      O  VMSYSU:KL.ADD.E
      P  VMSYSU:MAINT.ADD.PTF1
      Q  VMSYSU:MAINT.ADD
      R  VMSYSU:MAINT.ADD.NLL
      T  VMSYSU:DOST.TSTC2
      U  VMSYSU:KL.ADDT
      V  VMSYSU:MAINT.UTL
1= Help       2= Refresh  3= Quit   4= Sort(fm)    5= Sort(dir)   6= Auth
7= Backward   8= Forward  9=       10=            11= Filelist   12= Cursor

====>
                                                          X E D I T  1 File


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

Figure 2: Example output

DLALL EXEC

```
/*********************************************************************/
/* Calling DIRLIST for all directories of all filepools              */
/*********************************************************************/
/* Call:   DLALL <sort (Refresh>                                     */
/*         DLALL ?                                                   */
/*               sort = MODE          : sorts by filemode            */
```

```
/*               sort = DIR  (default): sorts by directory name       */
/*               Refresh              : investigate the new directory */
/*                                    :   structure                    */
/*                                    : otherwise the structure of     */
/*                                    :   the last DLALL call is       */
/*                                    :   displayed                    */
/*               ?                    : help                           */
/*********************************************************************/
trace off
parse upper arg sort '(' refresh
if sort = '?' then signal help
if sort ¬= '' & sort ¬= 'MODE' & sort ¬= 'DIR' then signal help
if sort ¬= '' & abbrev(refresh,'R') ¬= 1 then signal help
if sort = '' then sort = 'DIR'
'SET CMSTYPE HT'
'ESTATE' userid() 'DIRLALL A'
strc = rc
'SET CMSTYPE RT'
if abbrev(refresh,'R') = 1 | strc ¬= 0 then do
   'QUERY ENROLL USER FOR ALL VMSYSU: (STACK FIFO'
   anzuseru = queued()
   'QUERY ENROLL USER FOR ALL VMSYS: (STACK FIFO'
   anzuser = queued() - anzuseru
   do anzuseru
      parse upper pull user .
      if user = userid() then iterate
      if user = 'NUMBER' | user= '<PUBLIC>' then iterate
      queue 'DIRLIST VMSYSU:'user'. (APPEND ALL'
   end
   do anzuser
      parse upper pull user .
      if user = 'NUMBER' | user= '<PUBLIC>' then iterate
      queue 'DIRLIST VMSYS:'user'. (APPEND ALL'
   end
   if sort = 'MODE' then queue 'SOS PF4'
   if sort = 'DIR' then queue 'SOS PF5'
   queue 'SAVE'
   queue 'SET PF2 MSG Refresh bei DLALL nicht erlaubt'
   'DIRLIST VMSYSU:'userid()'. (ALL'
   address command 'COPY' userid() 'DIRLIST A = DIRLALL = (REPLACE'
end
else do
   address command 'COPY' userid() 'DIRLALL A = DIRLIST = (REPLACE'
   'DIRLIST (DIR' userid() 'ALL PROFILE DL'
end
exit
/*********************************************************************/
/* Help                                                              */
/*********************************************************************/
help:
```

```
'VMFCLEAR'
address cms 'type dlall   exec * 1 14'
```

PROCEDURE DL XEDIT

```
'MACRO PROFDLST'
'SET PF10 ?'
```

Dr Reinhard Meyer (Germany)

Repeated copying in a file

Often the same line needs to be populated in several places in a file. Since the standard prefix command 'C' clears itself after every single execution, you have to repeat the copy as many times as you want the line to be populated in the file. That's where this macro is useful. See Figure 1 below.

To enable the block prefix command, MCC, insert the following XEDIT command in PROFILE XEDIT:

```
command SET PREFIX SYNONYM MCC MC
```

EXEC

```
/*  'C' prefix command which allows for multiple 'F's or 'P's */
arg calltype callmode line
   If CallMode = 'CLEAR' then return    /* will erase itself         */
parse source . . myname . . calledName .

'command preserve'
'command Extract /LINE/'
  lineØ = line.1                        /* current line to restore   */
'command Set MsgMode Off'
If calledName = 'MCC'                   /* block command             */
 then do
  'command Locate :'line
```


Before	After	
==MC= if rc<>0 then	MC=== if rc<>0	Note that "MC" stays
=====	=====	ready for the next
=f=== 'EX R1' ->	===== 'EX R1'	invocation: scroll
===== 'EX R2'	===== if rc<>0	to the next page and
===p= line5	===== 'EX R2'	put subsequent "F"s
	===== if rc<>0	and "P"s there.
	=====	
=MCC= if rc<>0	MCC== if rc<>0	MCC is a block version
=MCC= then call er	MCC== then ...	of MC.
=f=== 'EX R1' ->	===== 'EX R1'	
===== 'EX R2'	===== if rc<>0	
===p=	===== then call er	
	===== 'EX R2'	
	===== if rc<>0	
	===== then call er	
	=====	

Figure 1: Use of the macro

```
  'command Set pending Off'
  'command Extract /PENDING MCC/'       /* closing block MCC         */
  'command Set Pending Block MCC'
    if pending.0=0 then signal ex       /* no closing block command  */
  mcc2 = pending.1
  'command Locate :'mcc2
  'command Set pending off'
  'command Set pending Block MCC'
  'command Down'
  'command extract /LINE/'
  ToLine = ':'Line.1                    /* copy up to this line      */
  blockmode=1
 end
 else do;blockmode=0;ToLine = 1
      end

Do forever                              /* process "F"s              */
  'command Extract /PENDING OLDNAME F/'
    If pending.0=0 then leave
  'command Locate :'line                /* 1st line to copy          */
```

```
  'command Copy' ToLine ':'pending.1    /* copy lines                */
  'command Extract /PENDING OLDNAME F/' /* line# could change if teof*/
  'command Locate :'pending.1
  'command Set Pending Off'             /* clear "F"                 */
end
Do forever                              /* process "P"s              */
  'command Extract /PENDING OLDNAME P/'
    If pending.0=0 then leave
  'command Locate :'pending.1;          /*P -prefixed line           */
  'command Set pending off'             /*clear pending              */
  'command UP'                          /*COPY inserts AFTER...      */
  'command Extract /LINE/'; lt=line.1   /*will copy after :lt        */
  'command Locate :'line                /*1st line to copy           */
  'command Copy' ToLine ':'lt
end

/* re-establish myself */
'command Locate :'line
If BlockMode then do
  'command Set Pending Block' CalledName
  'command Locate :'mcc2
  'command Set Pending Block' CalledName
  end
 else
  'command Set Pending on'    CalledName

ex:
'command Restore'
'command Locate :'line0
```

Vadim Rapp
Systems Officer
First Chicago NBD Corporation (USA)

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 Trevor Eddolls at any of the addresses shown on page 2. Why not call for a free copy of our *Notes for contributors* now?

CMS back-up/restore – part 2

This month we continue with the code for the CMS back-up/restore system.

```
NEXTUDEV EQU   *
         BAL   R14,GETBLKX             GO READ
         USING UDEVBLOK,R8
         AH    R8,DISP                 ADD DISPLACEMENT
         XC    UDEVTYPC(2),MASK        UNMASK DEVICE CLASS AND TYPE
         CLI   UDEVTYPC,X'Ø4'          IS IT A DASD ?
         BNE   SKIPVOL                 NO, IGNORE IT
         XC    UDEVVSER(6),MASK        UNMASK VOLUME ID
         CLC   REQVSER(4),ALLVSER      ALL VOLSERS REQUESTED
         BE    VOLSEROK                YES, BRANCH
         CLC   REQVSER(6),UDEVVSER     IS IT THE REQUESTED VOLSER
         BNE   SKIPVOL                 NO, IGNORE IT
VOLSEROK EQU   *
         MVC   STKLINE+14(6),UDEVVSER  MOVE VOLSER
         MVC   PACKCUU,UDEVADD         READY TO UNPACK CUU
         UNPK  UNPKCUU(5),PACKCUU(3)   UNPACK IT
         TR    UNPKCUU(4),HEXTBLP      TRANSLATE LETTERS
         MVC   STKLINE+9(4),UNPKCUU    AND MOVE RESULT
         XC    UDEVRELN(4),MASK        UNMASK START CYLINDER
         L     R1,UDEVRELN             ... AND LOAD IT
         BAL   R14,H2D                 ... THEN CHANGE TO DISPLAY
         MVC   STKLINE+21(4),TRANSR1   ... AND PLACE IT IN STKLINE
         XC    UDEVNCYL(4),MASK        DO THE SAME WITH NUMBER OF CYLS
         L     R1,UDEVNCYL             ...
         BAL   R14,H2D                 ...
         MVC   STKLINE+31(4),TRANSR1   ...
         L     R1,UDEVRELN             START CYLINDER
         L     R2,UDEVNCYL             NUMBER OF CYLINDERS
         BCTR  R2,Ø                    R2 = CYLINDERS - 1
         AR    R1,R2                   R1 = END CYL (START CYL + #CYLS)
         BAL   R14,H2D                 CONVERT
         MVC   STKLINE+26(4),TRANSR1   AND MOVE IT
         LA    R1,STACK                STACK THE ENTRY SO
         SVC   2Ø2                     THE EXEC CAN GET IT
         DC    AL4(OUT)                EXIT IF ERROR
SKIPVOL  EQU   *
         L     R2,UDEVDASD             GET DASD ADDRESS
         MVC   DISP,UDEVDISP           AND DISPLACEMENT
         LTR   R2,R2                   NO MORE
         BNZ   NEXTUDEV                YES, GO GET NEXT
         B     CHEKNDIR                ELSE GET NEXT USERID
UDIRCHEK EQU   *
         LTR   R2,R2                   END OF DIRECTORY
```

```
         BNZ   NEXTDIR                NO, GET NEXT DIRECTORY BLOCK
         B     OUT                    ELSE EXIT PROGRAM
         DROP  R8
*************************************************************************
**                E N D   O F   P R O G R A M                          **
*************************************************************************
OUT      EQU   *
         L     R13,SAVEAREA+4
         L     R14,12(R13)            RESTORE ADDRESSES
         LM    R0,R12,20(R13)         RESTORE ADDRESSES
         BR    R14
*************************************************************************
**                  S U B R O U T I N E S                              **
*************************************************************************
GETBLK1  EQU   *
         CL    R2,BLK1                DO WE ALREADY HAVE THE BLOCK
         BER   R14                    YES, JUST RETURN
         ST    R2,BLK1
         LA    R8,INBUFF1             PLACE DATA IN BUFFER 1
         B     GETBLK                 NOW GO GET IT
GETBLKX  EQU   *
         LA    R8,INBUFF2             ASSUME THE BLOCK IS IN BUFF2
         CL    R2,BLK2                DO WE
         BER   R14                    YES, JUST RETURN
         LA    R8,INBUFF3             ASSUME THE BLOCK IS IN BUFF3
         CL    R2,BLK3                DO WE
         BER   R14                    YES, JUST RETURN
         CL    R8,LASTBUFF            DID WE LAST USE BUFFER 3
         BE    USEBUFF2               YES, GO USE BUFFER 2
         ST    R2,BLK3                REMEMBER THIS BLOCK
         B     SAVEBUFF               GO SAVE USED BUFFER(3) ADDRESS
USEBUFF2 EQU   *
         ST    R2,BLK2                REMEMBER THIS BLOCK
         LA    R8,INBUFF2             USE BUFFER 2 FOR THE READ
SAVEBUFF EQU   *
         ST    R8,LASTBUFF            SAVE THE BUFFER ADDRESS
GETBLK   EQU   *
         STCM  R8,B'0111',BUFFADR     LET READ KNOW BUFFER ADDRESS
         STCM  R2,B'1100',CYL         STORE CYLINDER NUMBER
         STCM  R2,B'0010',RECORD      STORE  RECORD  NUMBER
         XR    R1,R1                  CLEAR R1 (HEAD NUMBER)
         XR    R2,R2                  CLEAR R2
         ICM   R2,B'0001',RECORD      SET R2 = RECORD NUMBER
HEADLOOP EQU   *
         CH    R2,TEN                 RECORD NUMBER > 10
         BNH   HEADOK                 NO, BRANCH
         SH    R2,TEN                 SUBTRACT 10 FROM RECORD NUMBER
         LA    R1,1(,R1)              AND ADD 1 TO HEAD NUMBER
         B     HEADLOOP               GO TRY AGAIN
HEADOK   EQU   *
         STCM  R1,B'0011',HEAD        STORE HEAD NUMBER
```

```
         ST    R14,BALSAVE             REMEMBER WHERE WE CAME FROM
         BAL   R14,READBLOK
         L     R1,LASTBUFF             LET R1 POINT TO THE BUFFER USED
         L     R14,BALSAVE             THIS IS WHERE WE CAME FROM
         BR    R14                     NOW GO BACK
*************************************************************************
READBLOK EQU   *
         LA    R15,1                   NUMBER OF READS IN CCW CHAIN
         DIAG  R4,R5,X'0018'           DO DASD I/O
         BCR   8,R14                   CC=0 - RETURN
         B     OUT                     ELSE GO END PROGRAM
*************************************************************************
H2D      EQU   *                       TRANSLATE R1 TO 4 DISPLAY DIGITS
         CVD   R1,DOBBWORD
         UNPK  TRANSR1(4),DOBBWORD+5(3)
         OI    TRANSR1+3,X'F0'
         BR    R14
*************************************************************************
**           C O N S T A N T S   A N D   V A R I A B L E S           **
*************************************************************************
DOBBWORD DS    D
SAVEAREA DS    18F
BALSAVE  DS    F
LASTSTRT DS    F
LASTBUFF DC    F'0'
NEXTUDIR DS    F
ONEPAGE  DC    F'4096'
CUU      DC    X'00000123'             DIRECTORY CUU
BLK1     DC    X'FFFFFFFF'
BLK2     DC    X'FFFFFFFF'
BLK3     DC    X'FFFFFFFF'
AND1     DC    X'00FF00FF'
MASK     DC    X'AAAAAAAAAAAAAAAA'
TEN      DC    H'10'
DISP     DS    H
         DS    0F
         DC    X'0000'
SEEK     DC    X'0000'
CYL      DC    X'0000'                 START WITH CYLINDER 0
HEAD     DC    X'0000'                             HEAD     0
RECORD   DC    X'03'                               RECORD   3
STACK    DS    0D                      PARAMETER LIST FOR THE CMS
         DC    CL8'ATTN'               FUNCTION: ATTN.
         DC    CL4'FIFO'
         DC    AL1(35)
         DC    AL3(STKLINE)
RDDISK   CCW   X'07',SEEK,X'40',6
         CCW   X'23',RECORD,X'40',1
         CCW   X'31',CYL,X'40',5
         CCW   X'08',*-8,X'00',0
LASTCCW  CCW   X'06',INBUFF1,X'00',4096
```

```
BUFFADR  EQU   LASTCCW+1
STKLINE  DC    CL35' '
TRANSR1  DS    CL4
REQVSER  DC    CL6' '
ALLVSER  DC    CL4'ALL '
PACKCUU  DS    CL2
         DC    XL1'ØF'                 DUMMY BYTE - IGNORED
UNPKCUU  DS    CL4
         DS    CL1                     DUMMY BYTE - IGNORED
HEXTBL   DC    X'ØAØBØCØDØEØF'
         DC    CL41' '
HEXTBL2  DC    C'Ø123456789ABCDEF'
HEXTBLP  EQU   HEXTBL-193
         LTORG
INBUFF1  DS    CL4Ø96
INBUFF2  DS    CL4Ø96
INBUFF3  DS    CL4Ø96
         DS    ØD
         COPY  UDIRECT
         END
```

BRMDSCAN ASSEMBLE

```
*********************************************************************
** BRMDSCAN - CMS Back-up/Restore mini-disk SCAN program.           **
** IF THE MINI-DISK IS AN (D)OS DISK, IT'S SELECTED WITHOUT FURTHER  **
** CHECKING                                                         **
*********************************************************************
BRMDSCAN CSECT
         EXTRN   BRVM2OS
         REGEQU
         STM     R14,R12,12(R13)
         USING   BRMDSCAN,R12,R9
         LR      R3,R13
         LA      R13,SAVEAREA
         ST      R13,8(R3)
         ST      R3,SAVEAREA+4
         LR      R9,R12
         A       R9,ONEPAGE
         CLC     8(8,R1),=CL8'GETLABEL'
         BNE     STARTPRG
         MVI     LABELGET,C'Y'
STARTPRG EQU     *
         FSSTATE FSCB=FSCBSCAN,ERROR=NOUFILE  CHECK FOR THE SCANLIST
         FSERASE FSCB=FSCBSEL          ERASE CMSBR SELECTED
         LA      R11,FSCBSEL
         USING   FSCBD,R11
         XC      FSCBNOIT,FSCBNOIT     SET NOREC=Ø
         DROP    R11
NEXTUSER EQU     *
```

```
         CLC     READPNT,H9
         BNH     MOVEINFO
         FSREAD  FSCB=FSCBSCAN          READ "CMSBR SCANLIST"
         CH      R15,RC12               END OF FILE
         BE      OUT                    YES, GO END PROGRAM
         LTR     R15,R15                RC = Ø
         BNZ     ERRUFILE               NO, GO STACK ERROR
         ST      RØ,BYTREAD
         XC      READPNT,READPNT
MOVEINFO EQU     *
         LH      R2,READPNT
         MH      R2,H57
         C       R2,BYTREAD
         BNL     OUT
         LA      R7,READBUFF(R2)
         USING   SCANLIST,R7
         LH      R2,READPNT             RELOAD COUNTER
         LA      R2,1(,R2)              ADD 1
         STH     R2,READPNT             STORE AGAIN
         BAL     R14,SETCUU             GO SET THE CUU
         MVC     BLKPOINT,=A(POINTBLK)
         MVC     READADDR(3),=AL3(INBUFF)
         MVC     LENGTH,=AL3(8Ø)        WE ONLY NEED 8Ø BYTES (1. READ)
         MVI     NOIGNORE,X'2Ø'         IGNORE INCORRECT LENGTH
         L       R2,REC3
         BAL     R14,READBLOK           GO DO FIRST READ (CCHHR = xxØØ3)
         CH      R15,RC13
         BE      NOFORMAT
         MVC     LABEL(6),ADTID         MOVE LABEL
         CLC     ADTIDENT(4),=C'VOL1'   IS IT A DOS/OS DISK
         BE      DOSDISK                YES, BRANCH
         MVC     DISKTYPE,CMS           MOVE " CMS " TO DISKTYPE
         CLI     LABELGET,C'Y'          ONLY THE LABEL WANTED
         BE      SELECT                 YES, JUST ACCEPT USER
         MVC     FSTSIZE,ADTFSTSZ       SAVE SIZE OF FST
         MVC     NFST,ADTNFST           SAVE NUMBER OF FST S
         MVC     LENGTH(3),ADTDBSIZ+1   SET LENGTH OF RECORD TO READ
         CLC     BLOCK8ØØ,ADTDBSIZ      IS BLOCK SIZE 8ØØ
         BE      DISK8ØØ                YES, BRANCH
         MVC     DATE(12),VARIES        MOVE DATE/TIME FOR PACK
         PACK    PDATE(7),DATE(13)      PACK DATE/TIME FOR COMPARE
         CP      PDATE(7),PACKØ         IS DATE/TIME = Ø
         BE      SELECT                 YES, JUST SELECT USER
         BAL     R14,VM2OS
         BAL     R14,INITREAD
         LA      R11,INBUFF
         USING   FSTD,R11
         CLC     FSTFNAME(16),DIRECTOR
         BNE     DMSGØØ5
         CLC     FSTAIC,FULL2           AIC = 2  (NO FILES ON DISK)
         BE      NEXTUSER               YES, GO GET NEXT USER
```

```
         CLC     PDATE(6),FSTADATI     UPDATED LATER THAN COMPARE
         BL      SELECT                YES, GO SELECT USER FOR BACK-UP
         B       NEXTUSER              ELSE GO GET NEXT USER
DISK800  EQU     *
         MVC     DATE800(4),MMDD       MOVE DATE (MMDD) FOR PACK
         MVC     DATE800+4(4),HHMM     MOVE TIME (HHMM) FOR PACK
         PACK    PDATE800(5),DATE800(9) PACK DATE/TIME FOR COMPARE
         CP      PDATE800(5),PACK0     IS DATE/TIME = 0
         BE      SELECT                YES, GO SELECT USER
         BAL     R14,VM2OS
         MVC     READADDR(3),=AL3(POINTBLK)
         MVC     ADTDOP(4),START800
         BAL     R14,INITREAD
         MVC     READADDR(3),=AL3(INBUFF)
         LA      R11,INBUFF
         USING   FSTD,R11
LOOP800  EQU     *
         L       R6,BLKPOINT           LOAD POINTBLK POINTER
         CLI     0(R6),X'FF'
         BE      NEXTUSER
         LH      R2,0(R6)
         LA      R6,2(,R6)
         ST      R6,BLKPOINT           SAVE POINTBLK POINTER
         BAL     R14,READBLOK          GO READ
         LTR     R15,R15
         BNZ     READERR
         L       R10,NFST              R10 = NUMBER OF FST'S PER BLOCK
         LA      R11,INBUFF
NEXT800  EQU     *
         CLI     0(R11),X'00'
         BE      NEXTUSER
         CLC     YEAR,FSTYEARW         IS LAST BACK-UP > UPDATED YEAR
         BH      IGN800                YES, GO GET ANOTHER FILE
         CLC     PDATE800,FSTDATEW     IS LAST BACK-UP >= UPDATED DATE
         BNH     SELECT                YES, ACCEPT MDISK FOR BACK-UP
IGN800   EQU     *
         A       R11,FSTSIZE
         BCT     R10,NEXT800
         B       LOOP800
DOSDISK  EQU     *
         MVC     DISKTYPE,DOSTEXT      MOVE "D(OS)" TO DISKTYPE
         B       SELECT                AND ACCEPT DISK FOR BACK-UP
NOFORMAT EQU     *
         MVC     LABEL,NA              MOVE N/A TO LABEL
         MVC     DISKTYPE,UNKNOWN      MOVE ? TO DISKTYPE
         B       SELECT                AND ACCEPT DISK FOR BACK-UP
SELECT   EQU     *
         LA      R2,VARIES
         MVC     VARIES(12),DISKINFO
         LA      R11,FSCBSEL
         USING   FSCBD,R11
```

```
         LH      R2,FSCBNOIT           LOAD RECORD COUNTER
         MH      R2,H53                CALCULATE OFFSET
         LA      R3,WRITEBUF(R2)       ADDRESS OF WRITEBUF + OFFSET
         MVC     Ø(53,R3),Ø(R7)        MOVE USERINFO INTO BUFFER
         LH      R2,FSCBNOIT           RELOAD RECORD COUNTER
         LA      R2,1(,R2)             MOVE COUNTER
         STH     R2,FSCBNOIT
         CH      R2,H9                 HAS MAXCOUNT BEEN REACHED
         BNE     NEXTUSER              NO, GO GET NEXT USER
         MH      R2,H53                CALCULATE BSIZE
         ST      R2,FSCBSIZE           STORE     BSIZE
         BAL     R14,WRITESEL
         XC      FSCBNOIT,FSCBNOIT     SET NOREC=Ø
         B       NEXTUSER              GO GET NEXT USER
*************************************************************************
**                    E N D   O F   P R O G R A M                      **
*************************************************************************
OUT      EQU     *
         LA      R11,FSCBSEL
         USING   FSCBD,R11
         LH      R2,FSCBNOIT           LOAD RECORD COUNTER
         LTR     R2,R2                 ANY RECORDS NOT WRITTEN
         BZ      CLOSE                 NO, GO CLOSE THE FILES
         MH      R2,H53                CALCULATE BSIZE
         ST      R2,FSCBSIZE           STORE     BSIZE
         BAL     R14,WRITESEL          WRITE THE LAST RECORDS
CLOSE    EQU     *
         FSCLOSE FSCB=FSCBSCAN         CLOSE FILES
         FSCLOSE FSCB=FSCBSEL
         L       R13,SAVEAREA+4
         L       R14,12(R13)           RESTORE ADDRESSES
         LM      RØ,R12,2Ø(R13)        RESTORE ADDRESSES
         L       R15,RC                SET RETURNCODE
         BR      R14
*************************************************************************
**                     M E S S A G E   S E T U P                       **
*************************************************************************
READERR  EQU     *
         MVC     MSGØØ1A(2),=C'18'
         B       DMSGØØ1
DIAG24E  EQU     *
         MVC     MSGØØ1A(2),=C'24'
DMSGØØ1  EQU     *
         LA      R3,MSGØØ1
         LA      R4,L'MSGØØ1
         B       ERROUT
SELERROR EQU     *
         LA      R3,MSGØØ2
         LA      R4,L'MSGØØ2
         B       ERROUT
NOUFILE  EQU     *
```

```
         LA      R3,MSG003
         LA      R4,L'MSG003
         B       ERROUT
ERRUFILE EQU     *
         LA      R3,MSG004
         LA      R4,L'MSG004
         B       ERROUT
DMSG005  EQU     *
         LA      R11,FSCBSEL
         USING   FSCBD,R11
         L       R15,FSCBRPTR
         SH      R15,READPNT
         LA      R3,MSG005
         LA      R4,L'MSG005
         B       ERROUT
DMSG006  EQU     *
         MVC     MSG006A(8),ULUSERID
         MVC     MSG006B(4),ULCUU
         LA      R3,MSG006
         LA      R4,L'MSG006
         B       ERROUT
ERROUT   EQU     *
         ST      R15,RC
         STCM    R3,B'0111',STKADDR
         STCM    R4,B'0001',STKSIZE
         LA      R1,STACK
         SVC     202
         DC      AL4(1)
         B       CLOSE
*************************************************************************
**                      S U B R O U T I N E S                          **
*************************************************************************
SETCUU   EQU     *
         PACK    DOBBWORD(8),ULSCYL(4)
         CVB     R3,DOBBWORD
         ST      R3,STARTCYL
         CLC     LASTCUU,ULVOLCUU
         BER     R14
         MVC     LASTCUU,ULVOLCUU
         PACK    CUU(3),LASTCUU(5)
         LH      R5,CUU
         DIAG    R5,R15,X'0024'
         BC      7,DIAG24E
         STCM    R15,B'1100',NEWTYPE
         BR      R14
*************************************************************************
WRITESEL EQU     *
         FSWRITE FSCB=FSCBSEL,ERROR=SELERROR
         BR      R14
*************************************************************************
** GET CAPACITY INFORMATION ABOUT THE DASD.                           ***
```

```
*************************************************************************
VM2OS    EQU     *
         CLC     DEVTYPE,NEWTYPE       SAME DEVICE TYPE AS LAST
         BNE     GETTYPE               NO, BRANCH
         CLC     BLKSIZE(2),ADTDBSIZ+2 SAME BLOCK SIZE AS BEFORE
         BER     R14                   YES, JUST RETURN
GETTYPE  EQU     *
         ST      R14,BALSAVE           STORE RETURN ADDRESS
         MVC     DEVTYPE,NEWTYPE       SAVE NEW DEVICE TYPE
         MVC     BLKSIZE(2),ADTDBSIZ+2 SAVE BLOCK SIZE INFO
         LA      R1,VMOSPARM           R1 -> PARM LIST FOR BRVM2OS
         L       R15,VMOS              LOAD AND ...
         BALR    R14,R15               RUN IT
         LTR     R15,R15               ERROR ?
         BNZ     DMSGØØ6               YES, BRANCH
         L       R14,BALSAVE           LOAD RETURN ADDRESS
         BR      R14                   RETURN
*************************************************************************
INITREAD EQU     *
         ST      R14,BALSAVE           STORE RETURN ADDRESS
         MVI     NOIGNORE,X'ØØ'        DO NOT IGNORE INCORRECT LENGTH
         MVC     CURRDIR,ADTDOP        REMEMBER ACTUAL DIRECTORY ADDR.
         L       R2,ADTDOP             SET UP R2 FOR READBLOK
         BAL     R14,READBLOK          GO READ
         LTR     R15,R15               READ ERROR ?
         BNZ     READERR               YES, TELL USER
         L       R1Ø,NFST              R1Ø = NUMBER OF FST S PER BLOCK
         L       R14,BALSAVE           RESTORE RETURN ADDRESS
         BR      R14                   RETURN
*************************************************************************
** R2 = RECORD NUMBER                                                  **
*************************************************************************
READBLOK EQU     *
         XC      SEEK(7),SEEK
         STCM    R2,B'ØØØ1',RECORD     ASSUME RECORD NUMBER IS OK
         C       R2,RECPRTRK           RECORD NUMBER > RECORDS/TRACK
         BNH     SEEKOK                NO, GO READ THE RECORD
         LR      R5,R2
         L       R2,RECPRTRK
         XR      R4,R4
         DR      R4,R2
         LTR     R4,R4
         BNZ     RECORDOK
         BCTR    R5,Ø
         L       R4,RECPRTRK
RECORDOK EQU     *
         STCM    R4,B'ØØØ1',RECORD     STORE  RECORD NUMBER
         STCM    R5,B'ØØ11',TRACK      ASSUME TRACK  NUMBER IS OK
         CH      R5,TRKPRCYL           TRACK NUMBER > TRACK/CYL
         BL      SEEKOK                NO, GO READ THE RECORD
         LH      R2,TRKPRCYL
```

```
         XR      R4,R4
         DR      R4,R2
         STCM    R4,B'0011',TRACK      STORE  TRACK   NUMBER
         STCM    R5,B'0011',CYL        STORE CYLINDER NUMBER
SEEKOK   EQU     *
         MVC     COMPARE,CYL
         XR      R5,R5
         ICM     R5,B'0011',CYL
         A       R5,STARTCYL
         STCM    R5,B'0011',CYL        STORE CYLINDER NUMBER
         LA      R15,1                 NUMBER OF READS IN CCW CHAIN
         LH      R4,CUU
         LA      R5,RDDISK
         DIAG    R4,R5,X'0018'
         BR      R14                   LET CALL CHECK RC
*********************************************************************
**          C O N S T A N T S   A N D   V A R I A B L E S          **
*********************************************************************
DOBBWORD DS      D
DIRECTOR DC      F'1',F'0',C'DIRECTOR'
SAVEAREA DS      18F
BALSAVE  DS      F
ONEPAGE  DC      F'4096'
RC       DC      F'0'
FULL2    DC      F'2'
REC3     DC      F'3'
START800 DC      F'4'
BLOCK800 DC      F'800'
STARTCYL DS      F
BYTREAD  DS      F
FSTSIZE  DS      F
NFST     DS      F
BLKPOINT DS      F
CURRDIR  DS      F                     ACTUAL DIRECTORY ADDRESS
VMOS     DC      V(BRVM2OS)
DEVSIZE  DS      0CL16
RECPRCYL DS      F                     RECORDS PER CYLINDER
RECPRTRK DS      F                     RECORDS PER TRACK
         DS      H
         DS      H
TRKPRCYL DS      H                     TRACKS  PER CYLINDER
         DS      H
         DS      0F
VMOSPARM DC      A(DEVSIZE)
BLKSIZE  DC      H'0'
DEVTYPE  DC      XL2'FF'
RC12     DC      H'12'
RC13     DC      H'13'
H9       DC      H'9'
H53      DC      H'53'
H57      DC      H'57'
```

```
READPNT  DC      H'1Ø'
CUU      DS      XL2,XL1
LASTCUU  DC      CL4'Ø',X'ØØ'
NA       DC      CL6'N/A'
UNKNOWN  DC      CL5'--?--'
DOSTEXT  DC      CL5'(D)OS'
CMS      DC      CL5' CMS '
NEWTYPE  DS      XL2
PACKØ    DC      PL1'Ø'
LABELGET DC      C'N'
DATE     DS      XL12
         DC      X'FØ'
PDATE    DS      XL6                  ONLY FIRST 6 BYTES NEEDED
         DS      XL1                  IGNORE THE LAST
DATE8ØØ  DS      XL8
         DC      X'FØ'
PDATE8ØØ DS      XL4                  ONLY FIRST 4 BYTES NEEDED
         DS      XL1                  IGNORE THE LAST
DISKINFO DS      ØCL12
DISKTYPE DS      CL5
         DC      CL1' '
LABEL    DS      CL6
         DS      ØF
         DC      X'ØØØØ'
SEEK     DC      X'ØØØØ'
CYL      DC      X'ØØØØ'              CYLINDER
TRACK    DC      X'ØØØØ'              TRACK
RECORD   DC      X'ØØ'                RECORD
COMPARE  DS      XL5
RDDISK   CCW     X'Ø7',SEEK,X'4Ø',6
         CCW     X'23',RECORD,X'4Ø',1
         CCW     X'31',COMPARE,X'4Ø',5
         CCW     X'Ø8',*-8,X'ØØ',Ø
LASTCCW  CCW     X'Ø6',INBUFF,X'ØØ',Ø
READADDR EQU     LASTCCW+1
NOIGNORE EQU     LASTCCW+4            X'2Ø' = IGNORE INCORRECT LENGTH
LENGTH   EQU     LASTCCW+5            IS SET DURING EXECUTION
*************************************************************************
**                          M E S S A G E S                            **
*************************************************************************
MSGØØ1   DC    C'Internal error - Error during DIAGnn'
MSGØØ1A  EQU   MSGØØ1+34
MSGØØ2   DC    C'Error writing CMSBR SELECTED to disk'
MSGØØ3   DC    C'No CMSBR SCANLIST file found'
MSGØØ4   DC    C'Error reading CMSBR SCANLIST file'
MSGØØ5   DC    C'Error during disk scan - No DIRECTOR record found.'
MSGØØ6   DC    C'Unsupported device. User: xxxxxxxx Cuu: xxxx.'
MSGØØ6A  EQU   MSGØØ6+26
MSGØØ6B  EQU   MSGØØ6+4Ø
*************************************************************************
**                        C M S   M A C R O S                          **
```

```
*************************************************************************
         DS    ØD
STACK    DC    CL8'ATTN'
         DC    CL4'LIFO'
STKSIZE  DC    AL1(Ø)
STKADDR  DC    AL3(Ø)
FSCBSCAN FSCB  'CMSBR SCANLIST A1',BUFFER=READBUFF,RECFM=F,NOREC=1Ø,   X
               BSIZE=57Ø
FSCBSEL  FSCB  'CMSBR SELECTED A1',BUFFER=WRITEBUF,RECFM=F,NOREC=Ø
         LTORG
WRITEBUF DS    CL53Ø
READBUFF DS    CL57Ø
         DS    ØF
         FSCBD
         DSECT
SCANLIST DS      ØCL57
SELECTED DS      ØCL53
ULUSERID DS      CL8
         DS      CL1
ULCUU    DS      CL4
         DS      CL1
ULVOLID  DS      CL6
         DS      CL1
ULSCYL   DS      CL4                  START CYLINDER
         DS      CL1
ULECYL   DS      CL4                  END   CYLINDER
         DS      CL1
ULTCYL   DS      CL4                  TOTAL CYLINDERS
         DS      CL1
         DS      CL4                  DASD TYPE
         DS      CL1
VARIES   DS      CL12
*                                     *** END OF SELECTED ***
ULVOLCUU DS      CL4
*                                     *** END OF SCANLIST ***
YEAR     EQU     VARIES
MMDD     EQU     VARIES+2
HHMM     EQU     VARIES+6
SS       EQU     VARIES+1Ø
BRMDSCAN CSECT
         DS    ØF
INBUFF   DS    CL4Ø96
ADTIDENT EQU   INBUFF                 LABEL  IDENTIFIER
ADTID    EQU   INBUFF+4               VOLUME IDENTIFIER
ADTDBSIZ EQU   INBUFF+12              DISK BLOCK SIZE
ADTDOP   EQU   INBUFF+16              ACTUAL DIRECTORY ADDRESS
ADTFSTSZ EQU   INBUFF+36              SIZE OF THE FST
ADTNFST  EQU   INBUFF+4Ø              NUMBER OF FST S PER BLOCK
POINTBLK DS    CL4Ø96
         DS    ØD
         FSTD
         END
```

BRVM2OS ASSEMBLE

```
*************************************************************************
** BRVM2OS - CMS Back-up/Restore VM -> OS device converter.           **
** ------------------------------------------------------------------ **
** R1 MUST POINT TO A LIST THAT CONTAINS:                            **
** F    ADDRESS WHERE TO RETURN INFO - THIS FIELD MUST BE 16C LONG   **
** H    DISK BLOCK SIZE                                              **
** X    DEVICE CLASS                                                 **
** X    DEVICE TYPE                                                  **
**                                                                   **
** RETURN CODES:                                                     **
** Ø    NO ERROR                                                     **
** 1    UNSUPPORTED DEVICE                                           **
**                                                                   **
** OUTPUT:                                                           **
** F    NUMBER OF RECORDS PER CYLINDER                               **
** F    NUMBER OF RECORDS PER TRACK                                  **
** H    PER ONE HEAD                                                 **
** H    OVERFLOW RECORD NUMBER (N/A)                                 **
** H    NUMBER OF TRACKS PER CYLINDER                                **
** H    MAXIMUM NUMBER OF CYLINDERS                                  **
*************************************************************************
BRVM2OS  CSECT
RØ       EQU   Ø
R1       EQU   1                       MUST POINT TO ADDRESS-LIST
R2       EQU   2                       WORK
R3       EQU   3                       WORK
R12      EQU   12                      BASE FOR BRVM2OS
R13      EQU   13
R14      EQU   14
R15      EQU   15                      RETURN CODE
         STM   R14,R12,12(R13)
         USING BRVM2OS,R12
         USING NUCON,RØ
         LR    R12,R15
         LR    R3,R13
         LA    R13,SAVEAREA
         ST    R13,8(R3)
         ST    R3,SAVEAREA+4
         LA    R15,1                   SET RC = 1 (ERROR)
*************************************************************************
** GET CAPACITY INFORMATION ABOUT THE DASD.                          ***
*************************************************************************
         CLI   6(R1),X'Ø4'             CKD DEVICE
         BNE   ERROR                   NO, EXIT WITH RC=1
         XR    R3,R3                   CLEAR R3
         IC    R3,7(R1)                R3 = VM DEVICE TYPE
         LTR   R3,R3
         BZ    ERROR
         A     R3,ADEVSUP              R3 = POINTER TO OS DEVTYPE CODE
```

```
         XR    R2,R2                   CLEAR R2
         IC    R2,Ø(R3)                R2 = VALUE AS POINTED BY R3
         A     R2,ADEVIND              R2 = INDEX DISPL. IN DWORDS
         XR    R3,R3                   CLEAR R3
         IC    R3,Ø(R2)                R3 = VALUE AS POINTED BY R2
         SLL   R3,3                    MULTIPLY BY 8  (WAS DWORDS)
         A     R3,ATBLIND              R3 = TRACK-CAPACITY INFO ADDR.
         ST    R3,DEVPOINT             STORE THE ADDRESS
         XR    R3,R3                   CLEAR R3
         XR    R2,R2                   CLEAR R2
         CLC   4(2,R1),BLOCK8ØØ        8ØØ BYTE BLOCK
         BE    DISK8ØØ                 YES, BRANCH
         ICM   R2,B'ØØ11',4(R1)        GET BLOCKSIZE
         SRL   R2,9                    SHIFT (SEE DMSDIP)
DISK8ØØ  EQU   *
         A     R2,ABLKIND              R2 = BLKSIZE INDEX POINTER
         IC    R3,Ø(R2)                R3 = BLKSIZE INDEX VALUE
         A     R3,DEVPOINT             R3 = DEVICE INFO ADDRESS POINTER
         L     R2,Ø(R3)                R2 = ADDRESS AS POINTED BY R3
         B     EXIT
*********************************************************************
**                   E N D   O F   P R O G R A M                   **
*********************************************************************
EXIT     EQU   *
         XR    R15,R15                 SET RC = Ø
         L     R3,Ø(R1)                R3 = WHERE TO PLACE RESULT
         MVC   Ø(16,R3),Ø(R2)          MOVE TRACK-CAPACITY INFORMATION
ERROR    EQU   *
         L     R13,SAVEAREA+4
         L     R14,12(R13)             RESTORE ADDRESS
         LM    RØ,R12,2Ø(R13)          RESTORE ADDRESSES
         BR    R14
*********************************************************************
**          C O N S T A N T S   A N D   V A R I A B L E S          **
*********************************************************************
SAVEAREA DS    18F
DEVPOINT DS    F
BLOCK8ØØ DC    X'Ø32Ø'
RC       DS    H
         NUCON
         END
```

BRDCNTRL XEDIT

```
/*********************************************************************
 ** BRDCNTRL  Back-up/Restore Display CoNTRoL - Controls:            **
 **                                                                  **
 ** the PF line                  Function: PF                        **
 ** the "running" line                     RUN                       **
 ** when to sound the alarm                ALARM                     **
```

```
** when to refresh display                REFRESH                  **
** reading from the screen                READ                     **
** lock all input fields                  NOINPUT                  **
**                                                                 **
** When RUN or ALARM is specified REFRESH is also executed.        **
**                                                                 **
** Separate functions with a % sign.                               **
*********************************************************************/
Parse Arg Input
Refresh = 'N'
Do While Input <> ' '
   Parse Value Input With Func Parms '%' Input
   Select
      When Func = 'PF' Then 'SET RESERVED -1 YEL NON HIGH Pf:' Parms
      When Func = 'RUN' Then Do
          'SET RESERVED 21 YEL NON HIGH',
               Center('***' Strip(Parms) '***',79)
          'CURSOR SCREEN 1 1'
           Refresh = 'Y'
           End
      When Func = 'REFRESH' Then Refresh = 'Y'
      When Func = 'ALARM'   Then Do
           Refresh = 'Y'
          'SOS ALARM'
           End
      When Func = 'NOINPUT' Then 'SET CTLCHAR ( PROTECT WHI NON NOHIGH'
      When Func = 'READ' Then Do
           Parse Value Parms With 'CURSOR=' Curline Curcol . 'KEYS=' .
           Parse Value Parms With 'KEYS=' Keys 'CURSOR='
           Do A=1 To 12 By 1
              If Find(Keys,A) <> Ø Then Value = A
                                   Else Value =
             'SET PF'A    VALUE
             'SET PF'A+12 VALUE
              End
           If Find(Keys,Ø) <> Ø Then 'SET ENTER BEFORE ENTER'
                                Else 'SET ENTER IGNORE'
           If Curline = ' ' Then Curline = 1
           If Curcol  = ' ' Then Curcol  = 1
           Read = 'Y'
           End
      Otherwise Nop
      End
   End
If Read    = 'Y' Then Do
  'CP SET EMSG TEXT'
   Call Read
  'CP SET EMSG OFF'
   Refresh = 'N'
   End
If Refresh = 'Y' Then Do
```

```
    Call Header
   'REFRESH'
    End
'SET CTLCHAR ( NOPROTECT WHI NON NOHIGH'
 Exit
/*********************************************************************
 ** HEADER DISPLAY                                                   **
 *********************************************************************/
Header:
'SET RESERVED 1 YEL NON H',
     Left(Date(),11) Center('CMS Back-up/Restore',53) Right(Time(),11)
 Return
/*********************************************************************
 ** READ FUNCTION                                                    **
 *********************************************************************/
Read:
 Call Header
'CURSOR SCREEN' Curline Curcol
'READ NO TAG'
 Parse Pull String
 Parse Value String With Key Num Value .
 If Key = 'CMD' Then Signal Read
 If Key = 'ETK' & Queued() > Ø Then Return
 If Key = 'PFK' Then String = 'PF'Value Num
 Push String
 Return
```

BRMAIN XEDIT

```
/*********************************************************************
 ** BRMAIN - CMS Back-up/Restore MAIN program.                       **
 *********************************************************************/
 Trace Off
 Signal On Syntax
 Call Init
 Pull Option BatchName
'DESBUF'
 Mdisk. = '?'
'EXEC BRQVDASD'
 If Rc = Ø Then Do
    Pull MDisk_List
    Do Queued()
       Pull Volser Mdisk.Volser
       End
    End
 If Option =  ' '       Then Signal MainMenu
 If Option =  'CHECK'   Then Signal MainMenu
 If Option =  'NOCLEAN' Then Signal MainMenu
 If Option <> 'BATCH'   Then Do
    Operinfo = 'Invalid option - 'Option
```

```
    Call Exit Ø
    End
/*********************************************************************/
/** BATCH mode specified                                            **/
/*********************************************************************/
'EXECIO * DISKR' BatchName 'BATCHRUN A 1 (FINIS STEM BATCH.'
 If Rc <> Ø Then Do
    Rx = Rc
    Operinfo = 'Invalid or missing BATCH name'
    Call Batchlog Ø Operinfo
    Call Exit Rx
    End
 Call Batchlog 1 'Batch input file:' BATCHNAME
 B_Nr    = Ø
 Cuu     = '???'
Batch:
 BHeader =
 Sleep   = Ø
 Run     = 'NO'
 TapeVol = 'BATCHØ'
 BInput. =
Batch_Loop:
 B_Nr = B_Nr + 1
 IF B_NR > BATCH.Ø THEN CALL EXIT Ø
 Parse Value Batch.B_Nr With Keyword Value .
 Select
    When Keyword = ' '         Then Nop
    When Keyword = '*'         Then Nop
    When Keyword = 'FUNCTION'  Then BInput.4  = Value
    When Keyword = 'ID'        Then BInput.6  = Value
    When Keyword = 'SORTLIST'  Then BInput.8  = Value
    When Keyword = 'LOGUPDATE' Then BInput.1Ø = Value
    When Keyword = 'TAPEVOL'   Then Tapevol   = Value
    When Keyword = 'TAPECUU'   Then Do
         CUU     = Translate(Word(Cp('QUERY VIRTUAL 181'),6))
         If CUU <> Value Then Do
            Parse Value Cp('DETACH 181') With .
            Parse Value Cp('ATTACH' Value '* 181') With Q_Rc CpText
            If Q_Rc <> Ø Then Do
               Call BatchLog Ø CpText 'RC=' Q_Rc
               Call Exit Q_Rc
               End
            End
         End
    When Keyword = 'PRINTLOG' Then Call Printlog
    When Keyword = 'ERASELOG' Then Do
        'ERASE CMSBR BATCHLOG A'
         BHeader = '1 Batch input file:' BATCHNAME
         End
    When Keyword = 'RUN' Then Run = 'YES'
    Otherwise
```

```
         Call Batchlog Ø 'Invalid keyword "'Keyword'" in line' B_Nr
         Call Exit Ø
    End
 IF RUN <> 'YES' THEN SIGNAL BATCH_LOOP
 Call Batchlog BHeader
 Call Batchlog 'Ø Batch run starting.  ' Date()',' Time()
 Call BatchLog '% FUNCTION ' BInput.4
 Call BatchLog '% ID       ' BInput.6
 Call BatchLog '% SORTLIST ' BInput.8
 Call BatchLog '% LOGUPDATE' BInput.1Ø
 Call BatchLog '% TAPECUU  ' Cuu
 Call BatchLog '% TAPEVOL  ' Tapevol
 Queue 'RES 4 33' BInput.4
 Signal MainMenu_X
/*********************************************************************
 ** Display available functions.                                    **
 *********************************************************************/
MainMenu:
 Line.   =
 If Option = 'BATCH' Then Do
    Call Batchlog '% Function' Binput.4 'completed.'
   'MACRO BRDCNTRL REFRESH'
    Signal Batch
    End
MainMenu_X:
 If Translate(Function) = 'RESTORE' Then,
    Parse Value BRTDISK('DETACH') With .
 Call Makeline 4,'Function',,9,'BF - Backup Full'
 Call Makeline 5,,,,'BL - Backup Limited'
 Call Makeline 6,,,,'RD - Restore Direct'
 Call Makeline 7,,,,'RI - Restore Indirect'
 Call Makeline 8,,,,'LT - ListTape'
 Call Makeline 9,,,,'LU - ListUser'
 Input.4 = Save.4
 If Operinfo <> ' ' Then Do
   'EMSG' Operinfo
    Operinfo =
    End
 PfLine = '1=Help  3=Exit'
 Cursor = '4 33'
MainMenu_Loop:
 If Input.4 <> ' ' Then Enter = 'OK'
                   Else Enter = 'IGN'
 Call Read
 If Act_Key = 'PF3' Then Call Exit Ø
 Save.4 = Input.4
 Parse Value Check(Input.4,4,'FUNCTION') With Reply
 If Pre.4   = 'Ø%' Then Do
    If Option = 'BATCH' Then Do
       Call BatchLog '% FUNCTION ' Help.4
       Call Exit Ø
```

```
      End
   Signal MainMenu_Loop
   End
 Parse Value Reply With Function Function_Parm Info.4
 Save4      = Line.4
 Line.      =
 Line.4     = Save4
 Pre.4      = 'Ø)'
 Accepted   = 'NO'
 Accept.NO  =
 Accept.YES = ALine('Input accepted, press ENTER to start ',
                    || Translate(Function))
 PfLine     = '1=Help  3=Return  4=Exit'
 Enter      = 'OK'
 Cur1       = 6
 Input.4    = Function
 Help.      = 'Must be specified.'
 If Option  = 'BATCH' Then Do
   If Function <> 'Backup' Then Do
      Call BatchLog '% FUNCTION  Can only be a BACK-UP function'
      Call Exit Ø
      End
   Help. = 'No error in field.'
   End
'FINIS CMSBR LOG A'
'ERASE CMSBR LOG A'
 Interpret('SIGNAL' Function)
```

Editor's note: this article will be continued next month.

Michael Plannthin (Denmark)

Subscribers who want copies of the code from this issue can call our Web site – http://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.

The old bulletin board service has now been discontinued because most people have Web access. We'd like to thank Ted Manos for all the work he did running our US bulletin board service for the past eight years.

Macro 4 has announced the availability of VSAMTUNE VSE Version 3.1, which introduces five new enhancements and extends the functionality available to VSAMTUNE VSE users.

For VM sites, the REXX/VSAM interface allows the automation of reporting and scheduling of jobs. VM, VSE and M4 REXX are all supported.

For further information contact:
Macro 4, The Orangery, Turners Hill Road, Worth, Crawley, W Sussex, RH10 4SS, UK.
Tel: (01293) 886060.
Macro 4, 35 Waterview Blvd, PO Box 292, Parsippany, NJ 07054-0292, USA.
Tel: (201) 402 8000.

* * *

IBM has announced Version 2.1 of its COBOL for VM and OS/390, adding to the COBOL object-oriented programming support on OS/390 that came out for the last MVS and VM version. Features include COBOL 85 standard language support, intrinsic functions, year 2000 support, inter-language communications, and the mainframe interactive debug tool.

Specifically, there's COBOL support for Dynamic Link Library (DLL) generation for OS/390 applications, and support for SOMobjects for OS/390 Release 3, including support for CORBA 2.0 object services and interoperability. The means of generating DLLs is similar to that used with OS/2, and the new support uses the same mechanisms as those used in OS/390 C/C++ and OS/390 SOMobjects, which means object-oriented COBOL applications can interact with them more easily. Support for SOMobjects for OS/390 Release 3, and for CORBA 2.0, is supposed to result in distributed applications that can take advantage of components residing on any platform using a CORBA compliant object request broker. SOMobjects will support naming, life cycle, persistence, security, externalization, and identity object naming services.

Also available is a COBOL Enterprise Workstation option, built around VisualAge for COBOL Professional for OS/2 Version 2.0. This includes a workstation development environment comprising compiler, run time, other development tools on OS/2, Windows 95, and Windows NT. It's also got a remote development capability, with edit, compile, and debug facilities, for working with OS/390 or MVS host applications from an OS/2 or NT workstation.

For further information contact your local IBM representative.

* * *

IBM's Tivoli has announced TME 10 NetView for OS/390. It combines functions found in NetView for MVS/ESA, NetView MultiSystem Manager, and Automated Operations Network/MVS (AON/MVS), into a single integrated product.

It's of interest to VM sites because it manages a heterogeneous network environment. And there's improved usability, including an enhanced version of NetView for OS/390 help and browse, as well as support for Language Environment for MVS and VM (LE/370).

For further information contact your local IBM representative.